BỘ GIÁO DỤC
VÀ ĐÀO TẠO

VIỆN HÀN LÂM KHOA HỌC
VÀ CÔNG NGHỆ VIỆT NAM

HỌC VIỆN KHOA HỌC VÀ CÔNG NGHỆ

PHẠM VĂN ĐĂNG

ĐẠI SỐ STREAM DÙNG CHO KHOA HỌC PHÂN TÍCH DỮ LIỆU LỚN DẠNG LIVESTREAM

LUẬN ÁN TIẾN SĨ MÁY TÍNH

Hà Nội – Năm 2025

BỘ GIÁO DỤC
VÀ ĐÀO TẠO

VIỆN HÀN LÂM KHOA HỌC
VÀ CÔNG NGHỆ VIỆT NAM

HỌC VIỆN KHOA HỌC VÀ CÔNG NGHỆ

PHẠM VĂN ĐĂNG

ĐẠI SỐ STREAM DÙNG CHO KHOA HỌC PHÂN TÍCH DỮ LIỆU LỚN DẠNG LIVESTREAM

LUẬN ÁN TIẾN SĨ MÁY TÍNH

Ngành: Khoa học máy tính

Mã số: 9 48 01 01

Xác nhận của Học viện
Khoa học và Công nghệ

KT. GIÁM ĐỐC
PHÓ GIÁM ĐỐC

HỌC VIỆN KHOA HỌC VÀ CÔNG NGHỆ
VIỆN HÀN LÂM KHOA HỌC VÀ CÔNG NGHỆ VIỆT NAM

Nguyễn Thị Trung

Người hướng dẫn 1
(Ký, ghi rõ họ tên)

PGS.TS.Phan Công Vinh

Người hướng dẫn 2
(Ký, ghi rõ họ tên)

TS.Trần Trọng Toàn

Hà Nội – Năm 2025

LỜI CAM ĐOAN

Tác giả xin cam đoan luận án: **"Đại số Stream dùng cho khoa học phân tích dữ liệu lớn dạng Livestream"** là công trình nghiên cứu của chính mình dưới sự hướng dẫn khoa học của Thầy PGS.TS.Phan Công Vinh và Thầy TS.Trần Trọng Toàn. Luận án sử dụng thông tin trích dẫn từ nhiều nguồn tham khảo khác nhau và các thông tin trích dẫn được ghi rõ nguồn gốc. Các kết quả nghiên cứu của tác giả được công bố chung với các tác giả khác đã được sự nhất trí của đồng tác giả khi đưa vào luận án. Các kết quả được trình bày trong luận án là hoàn toàn trung thực và chưa từng được công bố trong bất kỳ một công trình nào khác ngoài các công trình công bố của tác giả. Luận án được hoàn thành trong thời gian tác giả làm nghiên cứu sinh tại Học viện Khoa học và Công nghệ - Viện Hàn lâm Khoa học và Công nghệ Việt Nam, Viện Cơ học và Tin học Ứng dụng - Viện Hàn lâm Khoa học và Công nghệ Việt Nam.

Hà Nội, ngày 15 tháng 10 năm 2025

Tác giả luận án

(Ký và ghi rõ họ tên)

NCS.Phạm Văn Đăng

LỜI CẢM ƠN

Tác giả xin gửi lời cảm ơn chân thành đến Thầy PGS.TS.Phan Công Vinh và Thầy TS.Trần Trọng Toàn, Người hướng dẫn khoa học, Thầy ân cần, tận tâm, động viên, hướng dẫn và góp ý chuyên môn các công bố khoa học đăng trên các tạp chí, hội nghị, hội thảo trong nước và quốc tế uy tín cũng như cung cấp những tài liệu tham khảo quý báu để tác giả hoàn thành luận án này.

Tác giả xin chân thành gửi lời cảm ơn đến Thầy Cô của cơ sở đào tạo, Ban lãnh đạo, phòng Đào tạo, các phòng chức năng của Học viện Khoa học và Công nghệ - Viện Hàn lâm Khoa học và Công nghệ Việt Nam cũng như Đơn vị quản lý chuyên môn, Ban lãnh đạo, phòng Đào tạo, các phòng chức năng của Viện Cơ học và Tin học ứng dụng - Viện Hàn lâm Khoa học và Công nghệ Việt Nam đã giảng dạy, hướng dẫn, giúp đỡ, động viên, trang bị các kiến thức và tạo điều kiện tốt nhất cho tác giả để luận án được hoàn thành.

Tác giả xin gửi lời cảm ơn đến anh, chị, bạn bè, đồng nghiệp bằng nhiều hình thức khác nhau đã giúp đỡ tác giả trong quá trình học tập và nghiên cứu cũng như trong thời gian hoàn thành luận án này.

Đặc biệt, tác giả xin gửi lời tri ân sâu sắc đến Bố, Mẹ, Vợ, các con, cùng anh chị em trong gia đình – những người đã luôn ở bên động viên, chia sẻ và tiếp thêm sức mạnh tinh thần cho tác giả trong suốt những năm tháng thực hiện luận án. Luận án này, tác giả xin trân trọng dành tặng như một món quà tinh thần gửi đến toàn thể các thành viên trong gia đình.

Hà Nội, ngày 15 tháng 10 năm 2025

Tác giả luận án

(Ký và ghi rõ họ tên)

NCS.Phạm Văn Đăng

MỤC LỤC

DANH MỤC TỪ VIẾT TẮT

Stt	Từ viết tắt	Tên đầy đủ	Ý nghĩa
1	ASM	Algebraic semantics-based model	Mô hình dựa trên ngữ nghĩa đại số
2	ASA	Algebraic semantics-based automata	Máy trạng thái dựa trên ngữ nghĩa đại số
3	$\mathbb{A}^{\omega}$	$\mathbb{A}^{\omega}$	Tập các stream
4	ASA_{RTL}	Algebraic semantics-based RTL automata	Máy trạng thái mức chuyển đổi thanh ghi dựa trên ngữ nghĩa đại số
5	ASA_{SPEC}	Algebraic semantics-based specification automata	Máy trạng thái đặc tả dựa trên ngữ nghĩa đại số
6	BD	Big data	Dữ liệu lớn
7	BDL	Big data in livestream	Dữ liệu lớn dạng livestream
8	BA	Buffer average	Bộ đệm trung bình
9	BP	Buffer peak	Đỉnh bộ đệm
10	~	Bisimilarity	Tương đương hai chiều
11	CS	Computing system	Hệ thống tính toán
12	CStep	Control step	Bước điều khiển
13	Circle	Response composition	Liên kết phản hồi
14	Elevate	Elevating of homomorphism	Nâng cao tính đồng cấu
15	FU	Functional unit	Đơn vị chức năng
16	FBDL	Finite BDL	BDL hữu hạn
17	FSet	Finite sets of BDL	Tập các frame BDL hữu hạn
18	TFBDL	Timed finite BDL	BDL hữu hạn thời gian
19	TP	Throughput	Thông lượng
20	Hom	Homomorphism	Đồng cấu là một ánh xạ hoặc một hàm giữa hai cấu trúc đại số (như các automaton hoặc stream) mà bảo toàn các phép toán liên quan trong các cấu trúc đó.
21	HMonoid	Monoid homomorphism	Đồng cấu monoid
22	ICT	Information and Communication Technology	Công nghệ thông tin và truyền thông
23	IoMT	Internet of mobile things	Internet kết nối vạn vật di động
24	IH	Inductive hypothesis	Giả thiết quy nạp
25	IO	Input/Output	Nhập xuất
26	FI	Fairness index	Chỉ số đo cân bằng
27	KPI	Key performance indicator	Chỉ số hiệu năng chính
28	MWF	Monotonous witness function	Hàm chứng đơn điệu
29	PWF	Prefix witness function	Hàm chứng tiền tố
30	Para	Parallel composition	Liên kết song song hóa
31	LP-based PSOM	Linear programming-based pipeline scheduling optimization model	Mô hình tối ưu hóa lập lịch theo cơ chế đường ống dựa trên quy hoạch tuyến tính
32	LP	Linear programming	Quy hoạch tuyến tính
33	LA	Latency average	Độ trễ trung bình
34	LP	Latency p95	Độ trễ bách phân vị 95
35	RTL	Register transfer level	Mức chuyển đổi thanh ghi
36	Serial	Serial composition	Liên kết nối tiếp

Stt	Từ viết tắt	Tên đầy đủ	Ý nghĩa
37	9V	Volume	Khối lượng dữ liệu lớn
		Variety	Vận tốc dữ liệu lớn
		Velocity	Đa dạng dữ liệu lớn
		Veracity	Tính xác thực dữ liệu lớn
		Value	Tính chất giá trị dữ liệu lớn
		Variability	Tính biến thiên dữ liệu lớn
		Vision	Chỉ mục đích của phân tích dữ liệu lớn
		Verification	Xác định quá trình liên kết dữ liệu lớn
		Validation	Xác thực mục đích phân tích dữ liệu lớn
38	1C	Complexity	Chỉ sự rất khó tổ chức và phân tích dữ liệu lớn
39	1I	Immutability	Chỉ sự bất biến của dữ liệu lớn

DANH MỤC CỤM TỪ

Stt	Tên cụm từ	Ý nghĩa
1	Automata	Máy trạng thái
2	Batch	Bó/Lô
3	Bitstreams	Các luồng bit
4	Bisimulation	Mô phỏng hai chiều
5	Big data	Dữ liệu lớn
6	Big data analytics	Khoa học phân tích dữ liệu lớn
7	Combinators	Các bộ tổ hợp
8	Coinduction	Đồng quy nạp
9	Composition	Phép liên kết/Phép hợp thành
10	Big data stream	Luồng dữ liệu lớn
11	Counter of frames	Bộ đếm các frame
12	Counter of BDL	Bộ đếm BDL
13	Dropping operator	Phép toán loại bỏ (các luồng dữ liệu)
14	Frames	Các khung
15	Formalization framework	Khung hình thức hóa
16	Formalization	Hình thức hóa/Hình thức
17	Functor	Hàm tử (một cấu xạ từ phạm trù này sang phạm trù khác)
18	Formal variable	Biến hình thức
19	Feedback loop	Cơ chế hồi tiếp
20	Homomorphism	Đồng cấu
21	Information	Thông tin
22	Intelligence	Thông minh
23	Invariance	Sự bất biến
24	KB	Kịch bản
25	Livestream	Truyền trực tuyến
26	Merging operator	Phép toán gộp các luồng dữ liệu
27	Monotonous function	Hàm đơn điệu
28	Morphism	Cấu xạ
29	Multi-purpose FU	FU đa dụng

Stt	Tên cụm từ	Ý nghĩa
30	Modeling	Mô hình hóa
31	Non-standard stream calculus	Phép tính stream phi chuẩn
32	Parallelism composition	Liên kết song song hóa hoặc sự hợp thành song song hóa
33	Pipelining	Cơ chế đường ống
34	Response composition	Liên kết phản hồi (Một sự hợp thành phản ứng)
35	Reasoning	Phân tích tính chất/Lập luận
36	Stream (Infinite sequences)	Luồng (Dãy tuần tự vô hạn)
37	Scheduling	Lập lịch
38	Stream algebra	Đại số stream
39	Stream theory	Lý thuyết stream
40	Stream coalgebra	Đồng đại số stream
41	Stream calculus	Phép tính stream
42	Stream function	Hàm luồng (Hàm stream)
43	Stream equation	Phương trình luồng (Phương trình stream)
44	Stream differential equations	Các phương trình vi phân stream
45	Stream derivative	Đạo hàm stream
46	Sequences	Dãy tuần tự
47	Splitting operator	Phép toán tách các stream dữ liệu
48	Serial composition	Liên kết theo dạng nối tiếp hoặc sự hợp thành theo dạng chuỗi frame
49	Singleton	Một singleton là một tập hợp chỉ chứa đúng một frame. Nghĩa là, một tập hợp $\{x\}$ là một singleton nếu nó chỉ chứa đúng một frame x và không có frame nào khác.
50	Structured data: - Excel spreadsheets - Electronic forms - Data tables	Dữ liệu có cấu trúc - Bảng tính excel - Các dạng văn bản điện tử - Các bảng dữ liệu
51	Semi-structured data: - Email stores - JSON - NoSql - XML	Dữ liệu bán cấu trúc - Các dạng lưu trữ email - JSON - NoSQL - XML
52	Single-purpose FU	FU đơn dụng
53	Taking operator	Phép toán lấy các luồng dữ liệu
54	Formal verification	Kiểm chứng hình thức
55	Unstructured data: - Scanned PDFs - Text documents - Audio files - Video files - Images - Social media posts - Sensor data - Audio and Video	Dữ liệu phi cấu trúc - Dạng tập tin PDF được scan - Các văn bản - Các tập tin âm thanh - Hình ảnh, văn bản scan, nội dung đồ họa - Tweets, Facebook posts, các nội dung phát ra từ nhiều người dùng khác nhau - Dữ liệu từ các sensors: Các thiết bị IoT, và nó không có cấu trúc định nghĩa trước. - Recorded speeches, music, and video clips.
56	Unit delay	Độ trễ đơn vị
57	Zipping operator	Phép toán nén các stream dữ liệu

DANH MỤC BẢNG

DANH MỤC CÁC HÌNH VẼ VÀ SƠ ĐỒ

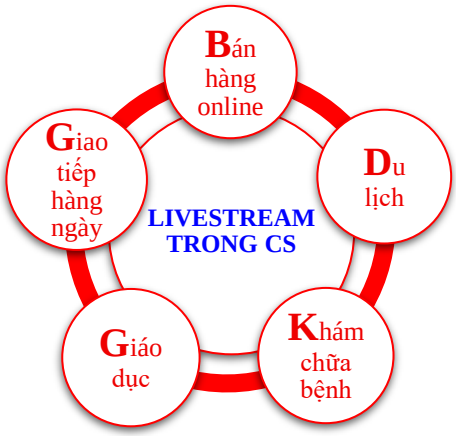
MỞ ĐẦU

1. Lý do chọn đề tài luận án

Trong bối cảnh hệ thống mạng xã hội và nền tảng số tạo ra lượng dữ liệu lớn (BD) gồm có dữ liệu lớn dạng Livestream (BDL) liên tục và có cấu trúc không đồng nhất, hiện tại tác giả luận án chưa tìm thấy một nền tảng toán học nào định nghĩa, giải thích và chứng minh về các cơ chế hoạt động của stream dạng BDL. Do đó, nhu cầu xây dựng một số khung lý thuyết hình thức cho BDL trở nên cấp thiết.

Vấn đề livestream là một trong những chủ đề đang được các nhà nghiên cứu về ICT quan tâm rất nhiều [1, 2]. Các hình thức Livestream phổ biến hiện nay hỗ trợ con người trong các ngành nghề: Livestream trong bán hàng [3, 4], Livestream trong du lịch [5, 6], Livestream trong khám chữa bệnh [7], Livestream trong giáo dục [8], và Livestream trong giao tiếp hàng ngày (Hình 1 và bảng 1), đa số các công trình [1-8] này xử lý livestream tập trung vào khía cạnh kỹ thuật, trong khi các mô hình toán học chặt chẽ để phân tích, biểu diễn và kiểm chứng tính đúng đắn của hoạt động livestream hầu như chưa được thiết lập. Ngoài ra, luận án cũng tiến hành khảo sát các nghiên cứu [9-21], các nghiên cứu này đã sử dụng lý thuyết stream để phân tích và tính toán stream. Lý thuyết stream chuẩn của Rutten [20] xử lý stream vô hạn nhưng không mô tả được sự biến thiên thời gian thực, thông lượng, độ trễ, cơ chế đường ống và bộ đệm (Đây là những đặc trưng thiết yếu của Livestream).

Luận án tập trung vào khảo sát, phân tích và nghiên cứu tập các stream ($\mathbb{A}^{\omega}$) dạng BDL. Livestream phát sinh ra lượng dữ liệu văn bản, hình ảnh, âm thanh và video rất lớn trong hệ thống tính toán (CS). Sự phát triển mạnh mẽ của livestream như vậy nhưng lại chưa có cơ sở lý thuyết toán học nào giải thích một cách hình thức về cơ chế hoạt động của livestream. Đây là cơ sở để luận án dựa vào lý thuyết stream (Stream theory) để phân tích, đặc tả, và hình thức hóa (Formalization) cơ chế hoạt động của stream dạng BDL (Hình 2) [CT.9] với tên đề tài luận án: **"Đại số Stream dùng cho khoa học phân tích dữ liệu lớn dạng Livestream"**.



Hình 1. Các hình thức livestream phổ biến tạo ra stream dạng BDL trong CS [CT.9]

Xây dựng một số khía cạnh đại số, lập lịch (Scheduling), và mô hình ngữ nghĩa đại số cho việc phân tích, đặc tả và hình thức hóa stream dạng BDL là hướng nghiên cứu chính của luận án. Với hướng nghiên cứu này, luận án đưa ra một số mở rộng nền tảng lý thuyết stream dùng cho khoa học phân tích stream dạng BDL bằng các định nghĩa, định lý, mệnh đề, hệ quả và các tính toán mở rộng cho stream tổ hợp thành các biểu thức stream nhằm hướng đến phân tích sâu hơn về stream dạng BDL.

Các stream dạng BDL liên tục được tạo ra trong thời gian thực bởi công nghệ livestream sẵn có trên các nền tảng mạng xã hội. Do đó, livestream được xem là một phương thức được sử dụng phổ biến hiện nay trong việc hỗ trợ con người ở các loại ngành nghề khác nhau được mô tả như sau (Bảng 1):

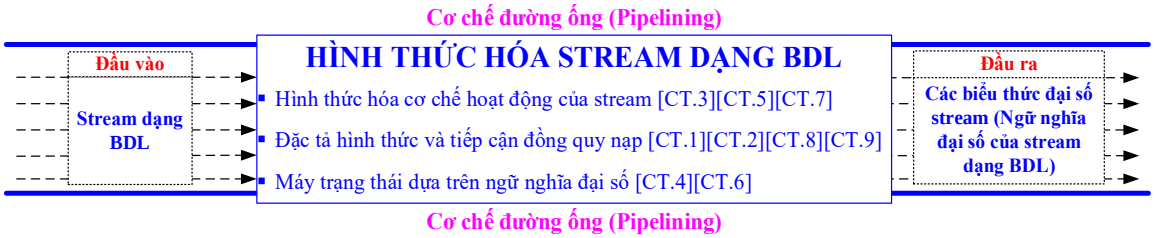
Bảng 1. Mô tả stream dạng BDL thông qua các hình thức livestream

Công trình	Năm	Hình thức livestream	Mô tả stream dạng BDL
[3, 22]	2021 2022	Livestream trong bán hàng	Stream dạng BDL được xem và mua hàng từ mua sắm trực tuyến của người dùng. BDL mua sắm trực tuyến giúp xác định các yếu tố ảnh hưởng đến ý định mua hàng, cung cấp trải nghiệm mua sắm thuận tiện, linh hoạt, giúp tương tác nhanh và giải trí hơn. Làm cơ sở cải thiện sự hài lòng và giúp nâng cao ý định sử dụng mua sắm trực tuyến liên tục của người tiêu dùng.
[5]	2022	Livestream trong du lịch	Stream dạng BDL điểm đến của các tour du lịch được công nghệ livestream phát trực tuyến sẽ luôn mang các đặc điểm ngay lập tức, tương tác chân thực và sống động tạo nên hình thái ảnh hưởng tích cực đến cảm giác hiện diện và sự tin tưởng của người tiêu dùng, tăng cường và ảnh hưởng trực tiếp đến ý định du lịch của khách hàng.
[7]	2021	Livestream trong khám chữa bệnh	Stream dạng BDL về cuộc phẫu thuật được công nghệ livestream phát trực tuyến trong thời gian thực qua luồng trực tuyến từ phòng mổ, hình ảnh được chuyển từ một kính hiển vi phẫu thuật kỹ thuật số và một máy ảnh môi trường ở chất lượng độ phân giải cao sẽ giúp học viên nắm bắt các kỹ thuật phẫu thuật rõ ràng và chính xác.
[7, 8]	2020 2021	Livestream trong giáo dục	Stream dạng BDL về bài giảng được tổ chức phát trực tuyến thông qua công nghệ livestream. BDL bài giảng được giảng viên tổ chức tạo ra các luồng công việc tùy chọn với các hoạt động cùng nhau tương tác giữa giảng viên và sinh viên bao gồm thăm dò ý kiến, hỏi, đáp, phản hồi tức thì và tạo nhóm tương tác.

2. Thách thức khoa học

Hình 2 mô tả tổng quan quá trình hình thức hóa cơ chế hoạt động của stream dạng BDL trong hệ thống livestream. Dữ liệu đầu vào là stream dạng BDL gồm có dãy các khung (Frames) được truyền tải liên tục thời gian thực và được xử lý qua cơ

chế đường ống (Pipelining).



Hình 2. Hình thức hóa cơ chế hoạt động của stream dạng BDL trong CS

Luận án đề xuất quá trình hình thức hóa gồm ba hợp phần: Hình thức hóa cơ chế hoạt động của stream nhằm xác định các đặc trưng động và trạng thái vận hành của BDL; Đặc tả cấu trúc stream và tiếp cận đồng quy nạp (Coinduction) trong việc mô hình hóa logic hoạt động của BDL; Xây dựng mô hình máy trạng thái (Automata) trên cơ sở ngữ nghĩa đại số, từ đó biểu diễn các hành vi hoạt động BDL.

Kết quả đầu ra của quá trình hình thức hóa này là ngữ nghĩa đại số của stream dạng BDL, đóng vai trò là đầu ra của hình thức hóa. Ngữ nghĩa đại số này được biểu diễn bằng các biểu thức đại số stream, dựa trên lý thuyết đại số, lý thuyết tập hợp và logic mệnh đề.

Dựa vào mô tả ở bảng 1, những phân tích của hình 2, và lý thuyết stream, luận án gặp phải những thách thức cho việc hình thức hóa stream dạng BDL như sau:

- Thách thức về tính sẵn có của hình thức hóa stream dạng BDL trong CS:
  - Thách thức về tính trừu tượng của mô hình hóa stream.
  - Thách thức về phân tích tính chất của mô hình hóa stream.
  - Thách thức về khả năng kiểm chứng hình thức stream.
- Thách thức về các tính chất tự nhiên sẵn có của stream: Tính liên tục, tính tốc độ cao, tính thay đổi, tính không đồng nhất, tính không hoàn chỉnh, tính ngắn hạn, và tính khối lượng lớn.

Tính liên tục của stream được tạo và truyền liên tục, không ngắt quãng. *Tính tốc độ cao* của stream sinh ra rất nhanh, yêu cầu xử lý thời gian thực. *Tính thay đổi* là nội dung và khối lượng thay đổi bất thường theo tình huống. *Tính không đồng nhất* của stream gồm nhiều frame định dạng như văn bản, hình ảnh, âm thanh, video. *Tính không hoàn chỉnh* của stream có thể thiếu hụt do tốc độ sinh dữ liệu và đa dạng nguồn. *Tính ngắn hạn* của stream chỉ có ý nghĩa trong thời gian ngắn, cần xử lý ngay. *Khối lượng lớn* là dữ liệu nhiều, gây áp lực cho lưu trữ và xử lý [CT.9].

3. Mục tiêu nghiên cứu

Mục tiêu nghiên cứu của luận án là xây dựng một số khía cạnh đại số, tối ưu hóa lập lịch, và mô hình ngữ nghĩa đại số (ASM) để hình thức hóa (mô hình hóa, phân tích tính chất, và kiểm chứng hình thức) cơ chế hoạt động của stream nhằm đáp ứng ngữ nghĩa đại số của stream dạng BDL trong các hệ thống tính toán (CS).

4. Nội dung nghiên cứu

Nội dung nghiên cứu 1: Tổng quan lý thuyết stream dùng cho phân tích dữ liệu lớn. Đề xuất các bảng phân tích và so sánh dựa trên các tiêu chí đặc trưng của lý thuyết stream có vai trò quan trọng trong phân tích BDL. Mở rộng đại số stream bằng mười phép toán trên stream và tổ hợp các phép toán này thành các biểu thức stream dạng BDL đã được công bố ở các công trình [CT.1][CT.2][CT.8].

Nội dung nghiên cứu 2: Nghiên cứu một số khía cạnh đại số của BDL trong IoMT, gồm: cấu trúc monoid, tính chất monoid, hình thức hóa BDL, phép biến đổi BDL, máy trạng thái và bộ tổ hợp máy trạng thái BDL. Một số khía cạnh đại số này cung cấp nền tảng lý thuyết giúp giải quyết vấn đề hình thức hóa stream dạng BDL đã được công bố ở các công trình [CT.3][CT.5][CT.7].

Nội dung nghiên cứu 3: Đặc tả hình thức và tiếp cận đồng quy nạp để kiểm chứng hình thức tính toán BDL qua thuật toán của phép tính stream (Stream calculus) trong tổng hợp các mức chuyển đổi thanh ghi (RTL). Nghiên cứu mối quan hệ giữa máy trạng thái mức chuyển đổi thanh ghi (ASA_{RTL}) và máy trạng thái đặc tả ngữ nghĩa đại số (ASA_{SPEC}) trong việc kiểm chứng hình thức stream BDL đã được công bố ở các công trình [CT.2][CT.4][CT.6][CT.9].

5. Ý nghĩa khoa học và thực tiễn của đề tài luận án

Ngày nay, BDL tồn tại phổ biến trong nhiều lĩnh vực của đời sống đòi hỏi phải có cơ sở lý thuyết và mô hình toán học cho BDL nhằm đưa ra các phương pháp xử lý và phân tích hiệu quả. Luận án phát triển các phép toán xử lý stream, cơ chế xử lý BDL dựa trên phát triển đại số stream kết hợp đồng đại số stream nhằm xây dựng cơ sở toán học cho BDL, trên cơ sở đó phát triển phương pháp phân tích và xử lý BDL.

6. Các đóng góp mới của luận án

- **Đóng góp 1:** Xây dựng một số khía cạnh đại số, gồm: Xây dựng mười phép toán xử lý stream, tổ hợp các phép này thành các biểu thức stream, mở rộng đại số stream và đồng đại số stream [CT.1][CT.2][CT.8], xây dựng cấu trúc monoid và tính chất monoid dùng cho stream dạng BDL [CT.3][CT.5][CT.7].
- **Đóng góp 2:** Xây dựng lịch biểu để xử lý stream dạng BDL gồm 04 bước sau: Phân hoạch các phép toán stream; Lập lịch cho tính toán BDL; Cấp phát và liên kết thanh ghi; Cấp phát và liên kết đơn vị chức năng [CT.2]; Tối ưu hóa lập lịch theo cơ chế đường ống [CT.9].
- **Đóng góp 3:** Xây dựng mô hình ngữ nghĩa đại số (ASM) để xử lý stream dạng BDL. Mô hình này cho phép đặc tả mối quan hệ giữa máy trạng thái mức chuyển đổi thanh ghi (ASA_{RTL}) và máy trạng thái đặc tả ngữ nghĩa đại số (ASA_{SPEC}) để kiểm chứng các kết quả tổng hợp RTL và SPEC [CT.4][CT.6].

7. Đối tượng và phạm vi nghiên cứu

7.1. Đối tượng nghiên cứu

Các stream, các frame, lý thuyết stream (đại số stream và đồng đại số stream), cùng với lý thuyết về mô hình hóa, phân tích tính chất và kiểm chứng hình thức.

7.2. Phạm vi nghiên cứu

Nghiên cứu lý thuyết stream tập trung các khía cạnh đại số, đặc tả hình thức và phương pháp đồng quy nạp nhằm kiểm chứng hình thức của stream dạng BDL.

8. Phương pháp nghiên cứu

- *Nghiên cứu lý thuyết:* Nghiên cứu tập trung vào việc tổng hợp và hệ thống hóa lý thuyết stream dùng cho stream, khảo sát chi tiết đặc điểm của chúng, đồng thời phân tích thế mạnh và hạn chế của đại số stream và đồng đại số stream. Trên cơ sở đó, luận án áp dụng các lý thuyết và phương pháp hình thức để chứng minh tính đúng đắn, phát triển và tổ hợp các phép toán nhằm xây dựng các biểu thức stream dạng BDL.
- *Nghiên cứu thực nghiệm:* Phần thực nghiệm được tiến hành thông qua việc đặc tả hình thức stream dạng BDL bằng các ví dụ minh họa và các kịch bản mô phỏng trên phép toán stream và thuật toán xử lý stream, qua đó minh chứng tính khả thi của mô hình lý thuyết. Đồng thời, các thuật toán được cài đặt dựa trên mô hình lý thuyết đề xuất để tối ưu hóa lập lịch theo cơ chế đường ống dựa trên phương pháp quy hoạch tuyến tính (LP-based PSOM [CT.9]).

9. Bố cục của luận án

Luận án được cấu trúc thành 04 chương theo bố cục như sau:

Mở đầu

Phần này giới thiệu tổng quan về luận án. Sau khi trình bày về sự phát triển của khoa học phân tích stream dạng BDL, luận án đã đúc kết những thách thức đối với stream dạng BDL. Từ những thách thức về BDL, luận án đặt ra mục tiêu để nghiên cứu.

Chương 1: TỔNG QUAN PHÂN TÍCH DỮ LIỆU LỚN DẠNG LIVESTREAM

Chương này tập trung trình bày tổng quan về phân tích dữ liệu lớn dạng livestream. Các phân tích và so sánh giúp luận án có hướng nhìn rõ nét về lý thuyết stream dùng cho khoa học phân tích stream.

Chương 2: XÂY DỰNG MỘT SỐ KHÍA CẠNH ĐẠI SỐ CỦA STREAM DẠNG BDL TRONG IoMT

Chương này trình bày về một số khía cạnh đại số của stream dạng BDL trong IoMT. Nền tảng lý thuyết này giúp đóng góp vào việc giải thích và chứng minh cơ chế hoạt động của stream dạng BDL.

Chương 3: ĐẶC TẢ HÌNH THỨC VÀ TIẾP CẬN ĐỒNG QUY NẠP ĐỂ KIỂM CHỨNG TỔNG HỢP TÍNH TOÁN BDL DỰA TRÊN PHÉP TÍNH STREAM

Chương này trình bày về mô hình hóa và tiếp cận đồng quy nạp để kiểm chứng hình thức tính toán stream dạng BDL. Nền tảng lý thuyết này chứng minh cơ chế hoạt động của stream.

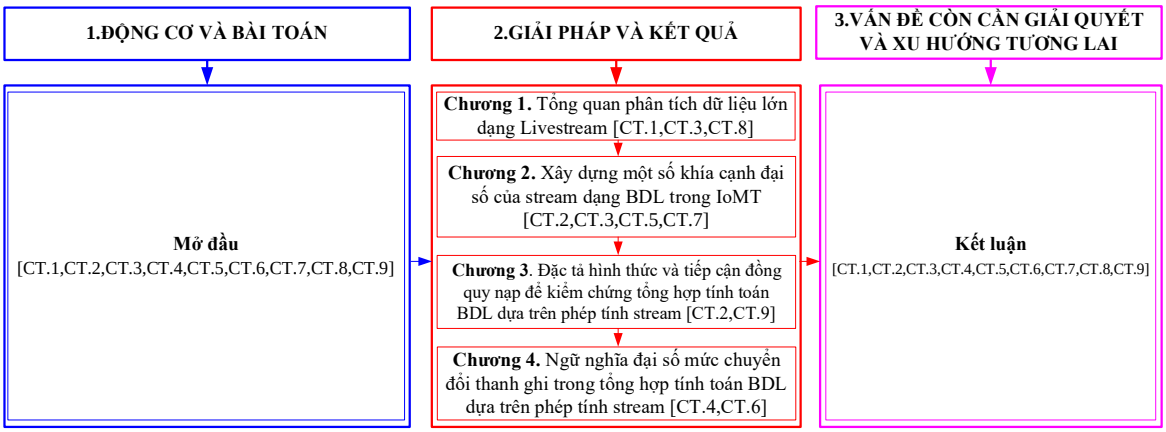
Chương 4: NGỮ NGHĨA ĐẠI SỐ MỨC CHUYỂN ĐỔI THANH GHI TRONG TỔNG HỢP TÍNH TOÁN BDL DỰA TRÊN PHÉP TÍNH STREAM

Chương này trình bày về mối quan hệ và ngữ nghĩa đại số mức chuyển đổi thanh ghi trong tổng hợp tính toán BDL dựa trên phép tính stream bằng các thuật toán có vòng lặp và không có vòng lặp.

Kết luận

Trong phần này, luận án tóm tắt các đóng góp khoa học và những nội dung tiếp tục được nghiên cứu.

10. Cấu trúc của luận án



Hình 3. Sơ đồ cấu trúc của luận án

11. Kết luận phần mở đầu

Phần mở đầu này đã tập trung giới thiệu tổng quan luận án. Sau khi trình bày về sự phát triển của khoa học phân tích stream dạng BDL, luận án đã đúc kết những thách thức đặt ra đối với stream dạng BDL. Từ những thách thức về stream dạng BDL, luận án đặt ra mục tiêu nghiên cứu và nội dung nghiên cứu. Nội dung chương 1 là trình bày về dữ liệu lớn, các công nghệ phân tích dữ liệu lớn, các khái niệm về lý thuyết stream (đại số stream và đồng đại số stream), các bảng phân tích và so sánh dựa trên các tiêu chí đặc trưng cũng như ưu điểm và hạn chế của lý thuyết stream trong phân tích stream dạng BDL trong các hệ thống tính toán.

Chương 1. TỔNG QUAN PHÂN TÍCH DỮ LIỆU LỚN DẠNG LIVESTREAM

Mục tiêu của chương 1 này phân tích xoay quanh dữ liệu lớn dạng livestream, các khái niệm cơ bản của lý thuyết stream, lập các bảng phân tích và so sánh dựa trên các tiêu chí đặc trưng của lý thuyết stream để nhìn thấy ưu điểm và hạn chế của lý thuyết stream trong phân tích stream dạng BDL. Từ đây, tác giả đặt ra 3 vấn đề nghiên cứu của luận án. Các kết quả phân tích của chương này được công bố trong công trình [CT.1] và [CT.8].

1.1. Giới thiệu

1.1.1. Phát sinh dữ liệu lớn

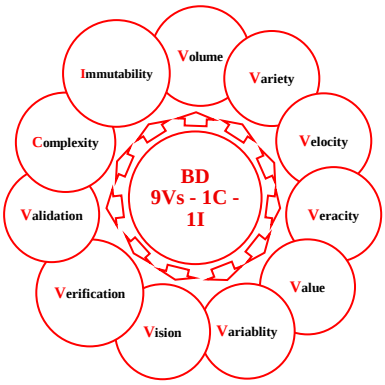
Dữ liệu là cốt lõi của mọi ngành nghề, thúc đẩy các chuyên gia ở nhiều lĩnh vực tham gia vào nghiên cứu: quy trình, thuật toán, mô hình toán học, nền tảng lý thuyết, hình thức hóa, mô hình hóa, phân tích tính chất và kiểm chứng hình thức [23]. Con người quan tâm dữ liệu lớn liên quan tới sự phức tạp nhiều hơn là kích thước, dữ liệu cần được phân tích và đặc tả cho phép các tổ chức giải quyết các bài toán phức tạp, giúp tổ chức ra quyết định và hành động tốt hơn.

Yếu tố quan trọng thúc đẩy khai thác dữ liệu lớn là các nền tảng tính toán ngày càng mạnh mẽ và ít tốn kém về khả năng tính toán lẫn lưu trữ. Áp dụng các công nghệ và kỹ thuật vào phân tích dữ liệu lớn livestream ngày càng trở nên phổ biến. Ngày nay nhiều người sử dụng: thiết bị thông minh, phương tiện truyền thông xã hội, hình thức livestream và các tài nguyên web, vậy họ đã dành lượng thời gian trực tuyến ngày càng nhiều làm phát sinh và tiêu thụ một lượng lớn dữ liệu [24-26].

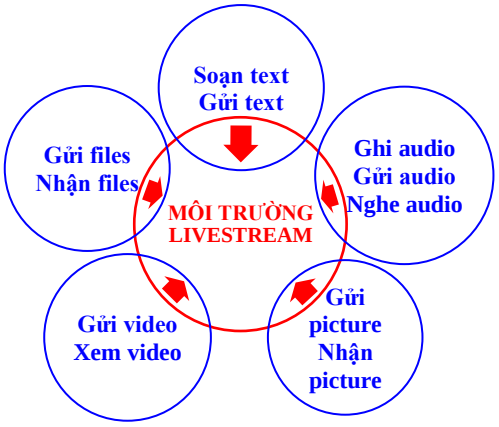
1.1.2. Một số tính chất đặc trưng của dữ liệu lớn

Một số thách thức kỹ thuật và công nghệ liên quan đến mười một tính chất đặc trưng của dữ liệu lớn: **V**olume, **V**elocity, **V**ariety, **V**eracity, **V**alue, **V**ariability, **V**ision, **V**erification, **V**alidation, **C**omplexity và **I**mmutability (9Vs - 1C - 1I) [27-29][CT.1] được khảo sát (Hình 1.1) và được mô tả vắn tắt như sau: **V**olume nghĩa là hỗ trợ dữ liệu rất lớn. **V**ariety nghĩa là đề cập đến các loại dữ liệu đang được lưu trữ gia tăng nhanh chóng theo thời gian thực. **V**elocity đề cập đến tốc độ mà dữ liệu mới phát sinh ra rất nhanh. **V**eracity đề cập đến mức độ tin cậy liên quan đến một số loại dữ liệu và mang ý nghĩa dữ liệu không chắc chắn. **V**alue với bản chất của dữ liệu là luôn luôn mang giá trị phục vụ cho nhiều loại lĩnh vực. **V**ariablity đề cập đến tính hỗn độn và không nhất quán của dữ liệu. **V**ision nghĩa là chỉ mục đích của việc phân tích dữ liệu. **V**erification nghĩa là xác định quá trình liên kết dữ liệu với một số các đặc tả. **V**alidation nghĩa là xác thực mục đích phân tích dữ liệu lớn phải được thực hiện. **C**omplexity nghĩa là chỉ rất khó tổ chức và phân tích dữ liệu lớn do các mối

quan hệ của dữ liệu lớn đang phát triển nhanh chóng. **I**mmutability chỉ sự bất biến (Invariance) của dữ liệu lớn, nghĩa là dữ liệu lớn được thu thập và lưu trữ có thể giữ nguyên không thay đổi vô thời hạn nếu được quản lý tốt.



Hình 1.1. Một số tính chất đặc trưng của dữ liệu lớn 9Vs – 1C – 1I



Hình 1.2. Các hoạt động livestream trên các nền tảng mạng xã hội

1.1.3. Hoạt động livestream

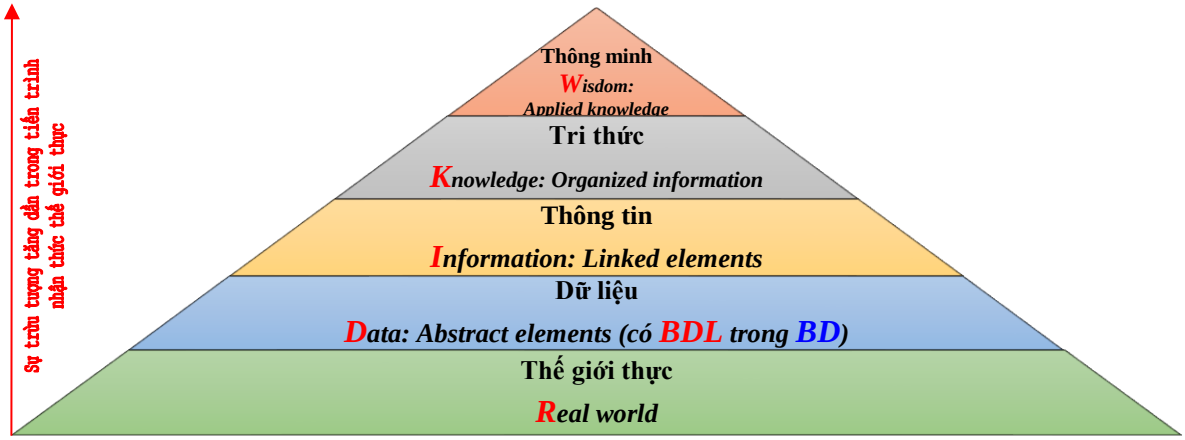
Phân tích stream dạng BDL là xử lý và phân tích các mẩu tin dữ liệu theo một cách liên tục thay vì bó các mẩu tin dữ liệu lại. Truyền trực tuyến được gọi là livestream, một hình thức phát trực tuyến stream phổ dụng và phổ biến ngày nay (Hình 1.2). Nói một cách tổng quát, phân tích stream dạng BDL hữu dụng cho các loại nguồn dữ liệu có cấu trúc (Structured data), dữ liệu bán cấu trúc (Semi-structured data), và dữ liệu phi cấu trúc (Unstructured data) mà nó truyền dữ liệu theo kích thước nhỏ dưới dạng kilobyte và dưới dạng stream liên tục trong thời gian thực.

Hoạt động livestream ngày nay trên các diễn đàn, youtube live, mạng xã hội rất phổ biến. Livestream được hiểu là tập hợp những hoạt động nhằm chuyển luồng các đa phương tiện diễn ra theo một cách trôi chảy. Trong môi trường livestream, các hoạt động như: *Soạn text, gửi text, ghi audio, gửi audio, nghe audio, gửi picture, nhận picture, gửi video, xem video, gửi files và nhận files* đang diễn ra rất phổ biến trong thời gian thực (Hình 1.2).

Sự phát triển của Internet đã thúc đẩy công nghệ livestream trở thành giải pháp hữu ích cho nhu cầu giải trí ngày càng cao, thay thế việc tải về các tập đa phương tiện nặng nề. Livestream là kỹ thuật truyền tải trực tiếp nội dung video qua các tập tin nén được chia thành các frame, cho phép người xem tua nhanh, tua chậm, tạm dừng hoặc tua lùi mà không cần tải về thiết bị. Quá trình này hoạt động bằng cách hệ thống tự động lưu và hiển thị stream tuần tự các frame trong bộ nhớ đệm khi phát trực tuyến. Stream các frame liên tục được tải và hiển thị cho đến khi kết thúc, tạo thành một luồng dữ liệu lớn (Big data stream) theo thời gian thực được gọi là hoạt động livestream stream dạng BDL.

1.1.4. Mối quan hệ của quá trình nhận thức tự nhiên

Hình 1.3, nhận thức thế giới thực là quá trình trừu tượng hóa, trong đó con người khảo sát dữ liệu *(Trong mối quan hệ của quá trình nhận thức tự nhiên lại có sự xuất hiện của BDL trong BD)*, biến đổi thành thông tin, rồi phân tích và khái quát thành tri thức về các đối tượng, sự kiện và hiện tượng. Mối quan hệ của quá trình nhận thức tự nhiên này giúp chúng ta tăng cường tiềm năng hỗ trợ ra quyết định nhiều lĩnh vực, BDL mang hình thái dạng BD có các phần tử trừu tượng.



Hình 1.3. Mối quan hệ của quá trình nhận thức **R** – **D** (**BDL**) – **I** – **K** – **W** tự nhiên

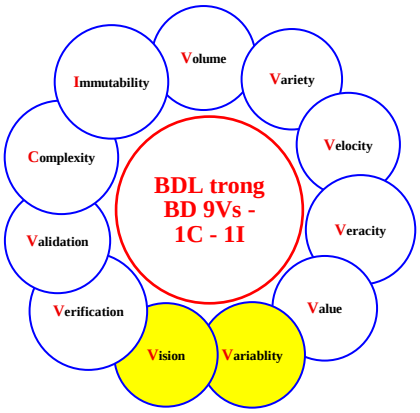
1.1.5. Khoa học phân tích dữ liệu lớn

Ngày nay, nhiều kỹ thuật: *khai thác dữ liệu*, *trực quan hóa dữ liệu*, *phân tích trực quan*, và *học máy* được áp dụng vào phân tích dữ liệu lớn. Nhiều nghiên cứu trong khoa học phân tích dữ liệu lớn (Big data analytics) bằng cách tăng cường các kỹ thuật đã sử dụng, đề xuất các kỹ thuật mới, hoặc thử nghiệm sự kết hợp của các thuật toán và công nghệ khác nhau. Chính dữ liệu lớn đã đẩy mạnh sự phát triển nhanh chóng của các kiến trúc hệ thống gồm cả phần cứng cũng như phần mềm.

Tuy nhiên, chúng ta rất cần những sự tiến bộ về phân tích dữ liệu để đối mặt với những thách thức về xử lý dữ liệu lớn và hình thức hóa stream phi cấu trúc dạng BDL phổ biến hiện nay. Một trong những vấn đề đặt ra làm sao nhận biết được cơ chế hoạt động của stream khi khối lượng BDL đang tăng lên theo thời gian [CT.8].

1.2. Các công nghệ phân tích dữ liệu lớn trong hệ sinh thái Hadoop

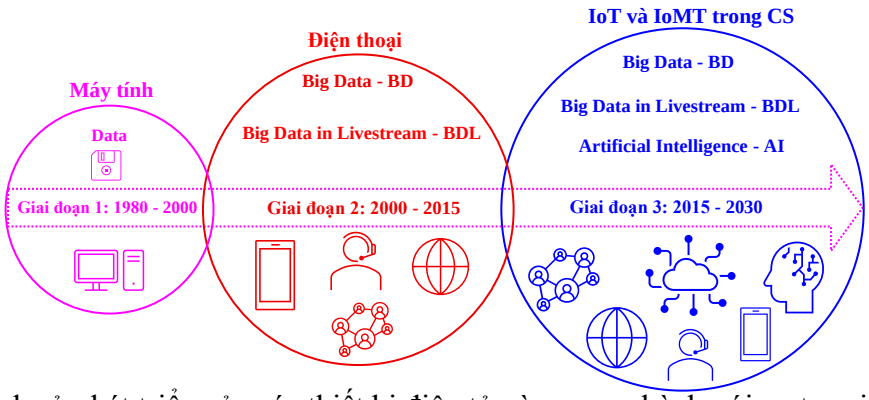
Hình 1.4. Các công nghệ dữ liệu lớn trong hệ sinh thái Hadoop



Hình 1.5. BDL trong BD có 9Vs – 1C – 1I [CT.1]

Theo khảo sát trong các công trình [27, 30-32] và bảng phân loại các công nghệ dữ liệu lớn trong hệ sinh thái Hadoop ở công trình [CT.1] đã phân tích và so sánh với nhau thông qua các tính chất đặc trưng của các công nghệ. Nhận thấy, hệ sinh thái này là một công nghệ dữ liệu lớn được chia thành năm tầng (Hình 1.4). Các công nghệ này cho phép phân tích dữ liệu từ các nguồn thu được từ cảm biến của IoT và IoMT.

1.3. Dữ liệu lớn dạng livestream



Hình 1.6. Lịch sử phát triển của các thiết bị điện tử cùng song hành với sự tạo sinh ra các nguồn dữ liệu lớn [CT.7]

Lịch sử phát triển của các thiết bị điện tử (PC, Telephone, IoT, IoMT) cùng song hành với sự tạo sinh ra các nguồn dữ liệu lớn qua các giai đoạn (Hình 1.6) [CT.7]. Những thách thức lý thuyết và kỹ thuật liên quan đến mười một tính chất đặc trưng 9Vs - 1C - 1I, BDL trong BD [CT.1] (Hình 1.5), BDL được xem là dãy tuần tự (Stream - Sequences) các frame tuyến tính và vô hạn. Một frame có thể được cấu trúc chứa các loại dữ liệu như: văn bản, hình ảnh, âm thanh, video, v.v. Nhận thấy, stream dạng BDL còn được gọi là dãy frame tuyến tính [CT.1][CT.2][CT.3].

1.4. Sự khác nhau và giống nhau giữa BD và BDL

Bảng 1.1. So sánh sự khác nhau và giống nhau giữa BD và BDL

BD (Big data)		Tính chất	BDL (Big data in livestream)	
♦		7**V**, 1**C**, 1**I**	♦	
Stream trừu tượng	✓	**V**ariablity	Stream biến đổi nhanh thời gian thực Stream lớn thời gian thực Stream có tính biến động Stream có tính xen kẽ các frame	✓
		Vision	Stream trực quan online Stream đáp ứng tính cảm nhận của thị giác Stream có tính biến động thời gian thực Stream đáp ứng tính ra quyết định	✓
Ghi chú: ✓ là thể hiện sự khác nhau; ♦ là thể hiện sự giống nhau				

1.5. Cơ chế đường ống trong xử lý stream dạng BDL

Cơ chế đường ống (Pipelining) là một chuỗi các tiến trình được thiết kế để tự động xử lý, tự động chuyển đổi và tự động lưu trữ stream dạng BDL trong thời gian thực từ nguồn đến đích, với các ứng dụng trong hệ thống IoT, IoMT và CS được phát biểu trong các công trình [CT.1][CT.2][CT.3][CT.9][33].

Ví dụ 1.1: Cho 4 stream a, b, c, d, e dạng $\mathbb{R}^{\omega}$ với biểu thức đại số: $e = (a + b) * (c - d)$. Giả sử mỗi phép toán ($+$, $-$, và $*$) là một công việc trong cơ chế đường ống và ta muốn lập lịch để các phép toán được thực hiện đúng logic phụ thuộc, tối ưu hóa chu kỳ thực thi, và tránh mối nguy cho dữ liệu [CT.9] (Bảng 1.2).

Bảng 1.2. Lập lịch theo cơ chế đường ống và so sánh với lập lịch không hợp lý

Lập lịch theo cơ chế đường ống (Pipelining)	So sánh với cách lập lịch không hợp lý
Bước 1: Xây dựng sự phụ thuộc cho các phép toán ▪ Các phép toán stream $\mathbb{R}^{\omega}$: $+$, $-$, $*$ ▪ Tạo sự phụ thuộc từ kết quả của $+$, $-$ đến $*$	Nếu cố gắng thực hiện phép toán $*$ trước phép toán $+$ hoặc phép toán $-$ (chưa hoàn thành) thì sẽ xảy ra mối nguy cho dữ liệu.
Bước 2: Lập lịch theo thứ tự phụ thuộc của các phép toán ▪ Những phép toán stream độc lập là: $+$, $-$ ▪ *Chu kỳ 1*, thực hiện phép toán (giai đoạn pipeline) $+$ và $-$ chạy song song. *Chu kỳ 2*, thực hiện $*$ sau khi $+$ và $-$ đã hoàn thành.	
Kết luận: ▪ Tìm thứ tự thực thi đúng dựa trên sự phụ thuộc dữ liệu của biểu thức đại số. ▪ Cho phép các phần độc lập được thực hiện song song.	

1.6. Lý thuyết nền tảng

Luận án tóm tắt lý thuyết stream (đại số stream – stream algebra và đồng đại số stream – stream coalgebra) làm cơ sở để chứng minh nền tảng lý thuyết này dành cho phân tích stream dạng BDL là hướng nghiên cứu của luận án.

Rutten [20] đề xuất một số ký hiệu và những kiến thức cơ bản về lý thuyết stream dùng cho phân tích dữ liệu như sau:

1.6.1. Các khái niệm cơ bản

a) Dữ liệu

Rutten định nghĩa các ký hiệu tập hợp như sau:

- $\mathbb{N}$: Tập hợp các số tự nhiên $0, 1, 2, 3, \ldots$
- $\mathbb{Z}$: Tập hợp các số nguyên $0, 1, -1, 2, -2, \ldots$
- $\mathbb{R}$: Các số thực $0.25, 0.1, 0.5, \ldots$
- $\mathbb{R}^{\omega}$: Tập các số thực $\{0.25, 0.1\}, \{0.5, 0.75, 0.25\}, \ldots$

b) Dữ liệu frame trong stream dạng BDL

Các công trình công bố của luận án [CT.1][CT.2] này đã định nghĩa các ký hiệu và các kiến thức cơ bản về stream dạng BDL như sau:

- Các mẫu tự Greek $\sigma, \tau, \rho, \alpha, \ldots$ được sử dụng để ký hiệu stream.

- Phần tử stream $\sigma \in \mathbb{A}^{\omega}$ được ký hiệu $\sigma = (\sigma(0), \sigma(1), \dots)$
- Đạo hàm của một stream σ là $\sigma' = (\sigma(1), \sigma(2), \dots)$
- Tập hợp các số tự nhiên $\mathbb{N} = \{0, 1, 2, \dots\}$
- Một số phép toán xử lý stream được xây dựng: dropping (drop), taking (take), zipping (zip), splitting (split), reverse (rev), merging (merge), convolution product ($\times$), copying ($\bullet$), registering ($\blacksquare$) và assignment ($:=$).
- $\mathbb{A}^{\omega}$: Tập các stream dạng BDL.
- $\mathbb{A}$: Tập các frame, một frame là một cấu trúc chứa các loại dữ liệu: văn bản (text), hình ảnh (image), âm thanh (audio), video, v.v.

c) Hàm và ảnh của hàm

Các hàm đôi khi cũng còn được gọi là các ánh xạ. Với một tập hợp X, hàm $1_X: X \to X$ và tính chất $1_X(x) = x$ được gọi là hàm đồng nhất. Với hai hàm $f: X \to Y$ và $g: Y \to Z$, Rutten [20] định nghĩa tính liên kết của f và g như sau: $g \circ f: X \to Z$ và $(g \circ f)(x) = g\big(f(x)\big)$. Liên kết $g \circ f$ được đọc là g sau f. Với hàm $f: X \to X$ thì ta định nghĩa $f^0 = 1_X$, còn nếu $f^{n+1} = f \circ f^n$.

Ảnh của hàm $f: X \to Y$ được định nghĩa bởi $image(f) = \{y \in Y \mid \exists x \in X, y = f(x)\}$. Cụ thể, với một tập con $V \subseteq X$, ảnh của V dưới hàm f được định nghĩa bởi $f(V) = \{y \in Y \mid \exists v \in V, y = f(v)\}$. Do đó, suy ra $image(f) = f(X)$. Đồ thị của hàm $f: X \to Y$ được định nghĩa bởi hàm $graph(f) = \{(x, y) \in X \times Y \mid y = f(x)\}$.

d) Phép toán tích và tổng

Tích Cartesian của hai tập hợp X và Y là $X \times Y = \{(x, y) \mid x \in X, y \in Y\}$. Phép chiếu của tích Cartesian $X \times Y$ có dạng $X \overset{\pi_1}{\leftarrow} X \times Y \overset{\pi_2}{\rightarrow} Y$, trong đó: $\pi_1\big((x, y)\big) = x$ và $\pi_2\big((x, y)\big) = y$. Tích của hai hàm $f_1: X_1 \to Y_1$ và $f_2: X_2 \to Y_2$ là $f_1 \times f_2: (X_1 \times X_2) \to (Y_1 \times Y_2)$, suy ra $(f_1 \times f_2)(x_1, x_2) = \big(f_1(x_1), f_2(x_2)\big)$. Ghép cặp hai hàm $f: Z \to X$ và $g: Z \to Y$ là $\langle f, g \rangle: Z \to X \times Y$, $\langle f, g \rangle(z) = \big(f(z), g(z)\big)$.

Tổng của hai tập hợp X và Y là $X + Y = X \uplus Y$, ở đó hợp tách rời *(Hợp không giao nhau* $\uplus$*, với* $(x, 0)$ *là phần tử từ tập X được gắn nhãn* 0*, và* $(y, 1)$ *là phần tử từ tập Y được gắn nhãn* 1*)* được định nghĩa $X \uplus Y = \{(x, 0) \mid x \in X\} \uplus \{(y, 1) \mid y \in Y\}$. Các ánh xạ nhúng của $X + Y$ là $X \overset{\kappa_1}{\rightarrow} X + Y \overset{\kappa_2}{\leftarrow} Y$, trong đó $\kappa_1(x) = (x, 0)$ và $\kappa_2(y) = (y, 1)$. Tổng của $f_1 + f_2: (X_1 + X_2) \to (Y_1 + Y_2)$ của hai hàm f_1 và f_2 là $f_1: X_1 \to Y_1$ và $f_2: X_2 \to Y_2$ được định nghĩa như sau: $(f_1 + f_2)(x_1, 0) = (f_1(x_1), 0)$ và $(f_1 + f_2)(x_2, 1) = (f_2(x_2), 1)$. Đồng ghép cặp hai hàm $f: X \to Z$ và $g: Y \to Z$ có dạng $[f, g]: (X + Y) \to Z$ được định nghĩa như sau $[f, g]\big((x, 0)\big) = f(x)$ và $[f, g]\big((y, 1)\big) = g(y)$.

e) Tập hợp con

Tập hợp tất cả các tập con của một tập X được ký hiệu là $\mathcal{P}(X) = \{V \mid V \subseteq X\}$. Tập hợp $\mathcal{P}(X)$ được gọi là lũy thừa của tập X. Tập rỗng không chứa phần tử nào được ký hiệu là $\emptyset$. Đối với $\mathcal{V} \subseteq \mathcal{P}(X)$, Rutten [20] định nghĩa phép toán hợp và phép toán giao của $\mathcal{V}$ như sau $\bigcup \mathcal{V} = \{x \in X \mid \exists V \in \mathcal{V}, x \in V\}$ và $\bigcap \mathcal{V} = \{x \in X \mid \forall V \in \mathcal{V}, x \in V\}$.

f) Quan hệ

Một quan hệ giữa hai tập hợp X và Y là một tập con $R \subseteq X \times Y$. Phép chiếu của R là $X \overset{\pi_1}{\leftarrow} R \overset{\pi_2}{\rightarrow} Y$, trong đó $\pi_1\big((x,y)\big) = x$ và $\pi_2\big((x,y)\big) = y$. Nghịch đảo của $R \subseteq X \times Y$ là $R^{-1} \subseteq Y \times X$ suy ra $R^{-1} = \{(y,x) \in Y \times X \mid (x,y) \in R\}$. Hợp của hai quan hệ $R \subseteq X \times Y$ và $S \subseteq Y \times Z$ là $R \circ S = \{(x,z) \in X \times Z \mid \exists y \in Y, (x,y) \in R$ và $(y,z) \in S\}$ được đọc là R theo sau bởi S. Một quan hệ $R \subseteq X \times X$ là một quan hệ tương đương nếu nó thỏa mãn các tính chất: $x \leq x$ là tính chất phản xạ, $(x \leq y$ và $y \leq z) \Longrightarrow x \leq z$ là tính chất bắc cầu, và $x \leq y \Longrightarrow y \leq x$ là đối xứng.

Với một quan hệ tương đương R trên X và $x \in X$, lớp tương đương theo R của x là $[x]_R = \{y \in X \mid (x,y) \in R\}$. Tập thương của X theo quan hệ tương đương R là $X/R = \{[x]_R \mid x \in X\}$, tức là tập hợp các lớp tương đương theo R. Hàm thương của R là $q\colon X \to X/R$, trong đó $q(x) = [x]_R$.

g) Cấu trúc monoid

Định nghĩa 1.1 [20, 34, 35]: Cấu trúc monoid là một bộ ba $(M, \cdot, e)$, trong đó: M là một tập hợp hữu hạn hoặc vô hạn, $\cdot$ là một phép tính stream, $e \in A$ là phần tử đơn vị sao cho thỏa mãn $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ và $1 \cdot a = a \cdot 1 = a, \forall a, b, c \in M$.

h) Quan hệ thứ tự và thứ tự cục bộ

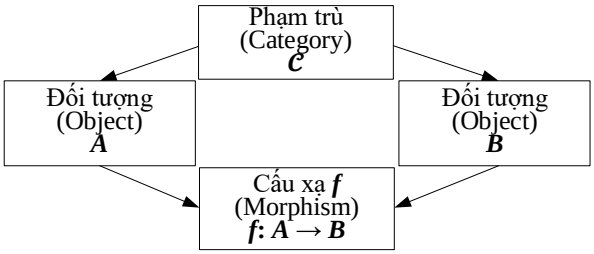
Một quan hệ thứ tự có dạng $(P, \leq)$, trong đó $\leq \subseteq P \times P$ gồm có một tập hợp P và một quan hệ hai ngôi $\leq$ thỏa mãn hai tính chất sau đây: $a \leq a$ có tính phản xạ, $a \leq b$ và $b \leq c \Longrightarrow a \leq c$ có tính chất bắc cầu. Nếu một quan hệ tiền thứ tự thỏa mãn tính chất $(a \leq b)$ và $(b \leq a) \Longrightarrow a = b$ có tính chất phản đối xứng thì nó là quan hệ thứ tự cục bộ. Mặt khác, nếu một quan hệ tiền thứ tự thỏa mãn tính chất $(a \leq b) \Longrightarrow (b \leq a)$ có tính chất đối xứng, khi đó $\leq$ là một quan hệ tương đương. Giả sử $(P, \leq)$ và $(Q, \leq)$ là hai quan hệ tiền thứ tự. Nếu hàm f có dạng $f\colon P \to Q$ thỏa mãn tính chất sau $(a \leq b) \Longrightarrow f(a) \leq f(b)$ thì hàm f là đơn điệu.

i) Cấu xạ với phạm trù

Cấu xạ (Morphism) là một khái niệm trừu tượng, và hàm và đồng cấu (Homomorphism) là những trường hợp cụ thể của cấu xạ, tùy thuộc vào phạm trù cụ thể thì ý nghĩa của cấu xạ có thể khác nhau [20]:

- Nếu phạm trù là tập hợp, thì cấu xạ f là một hàm.
- Nếu phạm trù là nhóm, thì cấu xạ f là một đồng cấu nhóm.

- Nếu phạm trù là không gian Topo, thì cấu xạ f là một hàm liên tục.



Hình 1.7. Sơ đồ phạm trù với cấu xạ của hai đối tượng A và B

Trong một phạm trù là tập hợp Set, một cấu xạ là một ánh xạ từ một đối tượng này đến một đối tượng khác. Nếu A và B là hai đối tượng trong phạm trù Set, thì một cấu xạ f từ A đến B được ký hiệu là $f: A \to B$ (Hình 1.7). Một phạm trù gồm:

- Tập các đối tượng
- Tập các cấu xạ giữa các đối tượng
- Phép liên kết (Composition) giữa các cấu xạ: Nếu $f: A \to B$ và $g: B \to C$, thì tồn tại phép liên kết $g \circ f: A \to C$
- Cấu xạ đơn vị: Với mỗi đối tượng A, tồn tại $\mathrm{id}_A: A \to A$, sao cho $f \circ \mathrm{id}_A = f$, $\mathrm{id}_B \circ f = f$, với mọi $f: A \to B$

j) So sánh cấu xạ với hàm và với đồng cấu

Bảng 1.3. Bảng so sánh cấu xạ với hàm và với đồng cấu

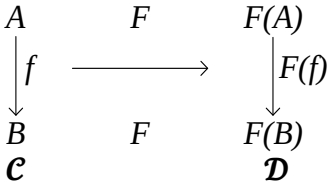
Thuộc tính	Cấu xạ	Hàm	Đồng cấu
Nơi xuất hiện	Lý thuyết phạm trù	Toán học nói chung, đặc biệt trong tập hợp	Đại số trừu tượng (nhóm, vành, không gian vector)
Ý nghĩa	Ánh xạ trừu tượng giữa hai đối tượng	Một quy tắc ánh xạ phần tử từ tập này sang tập khác	Hàm bảo toàn cấu trúc đại số (nhóm, vành, v.v)
Ký hiệu	$f: A \to B$	$f(x) = y$ với $x \in A$ và $y \in B$	$f(a * b) = f(a) \cdot f(b)$ với với $f(*) = \cdot$
Bảo toàn cấu trúc?	Không nhất thiết	Không	Có (Điều kiện bắt buộc)
Tính chất bắt buộc	Có liên kết và đơn vị (Trong phạm trù)	Không bắt buộc	Phải thỏa mãn tính bảo toàn cấu trúc

k) Hàm tử

Định nghĩa 1.2 [20]: Một hàm tử F là một cấu xạ giữa hai phạm trù, ký hiệu: $F: \mathcal{C} \to \mathcal{D}$, trong đó $\mathcal{C}$ và $\mathcal{D}$ là hai phạm trù. Hàm tử (Functor) làm hai nhiệm vụ: *(1) Ánh xạ các đối tượng*, với mỗi đối tượng $A \in \mathcal{C}$, ánh xạ thành một đối tượng $F(A) \in \mathcal{D}$; *(2) Ánh xạ các cấu xạ*, với mỗi cấu xạ $f: A \to B$ trong phạm trù $\mathcal{C}$, ánh xạ thành một cấu xạ $F(f): F(A) \to F(B)$ trong phạm trù $\mathcal{D}$ (Hình 1.8).

Do vậy, hàm tử phải thỏa mãn hai điều kiện như sau: *(1) Bảo toàn tính liên kết:* $F(g \circ f) = F(g) \triangle F(f)$, với $f: A \to B$, $g: B \to C$, và $F(\circ) = \triangle$; *(2) Bảo toàn tính*

cấu xạ đơn vị: $F(\mathrm{id}_A) = \mathrm{id}_{F(A)}$.



Hình 1.8. Sơ đồ thể hiện cách hàm tử F ánh xạ từ phạm trù $\mathcal{C}$ sang phạm trù $\mathcal{D}$

l) Hàm tử trên các tập hợp

Hàm tử là một cấu xạ từ phạm trù này sang phạm trù khác. Phạm trù Set bao gồm các tập hợp và các hàm. Phạm trù Set × Set có các cặp tập hợp X_1 và X_2 là các đối tượng và các cặp hàm $f_1: X_1 \to Y_1$ và $f_2: X_2 \to Y_2$ là các ánh xạ, trong đó đối tượng ám chỉ (X_1, X_2) hoặc (Y_1, Y_2), còn ánh xạ ám chỉ $(f_1, f_2): (X_1, X_2) \to (Y_1, Y_2)$. Các hàm tử $F_\times: Set \times Set \to Set$ và $F_+: Set + Set \to Set$ được định nghĩa trên phép toán cộng và phép toán tích của các tập hợp và hàm như được định nghĩa trong *mục d của mục 1.6.1* được phân tích trong bảng 1.4 như sau:

Bảng 1.4. Phép toán tích và tổng các đối tượng và các ánh xạ trong hàm tử.

Các đối tượng	Các ánh xạ
$F_\times((X_1, X_2)) = X_1 \times X_2$	$F_\times((f_1, f_2)) = (f_1 \times f_2): (X_1 \times X_2) \to (Y_1 \times Y_2)$
$F_+((X_1, X_2)) = X_1 + X_2$	$F_+((f_1, f_2)) = (f_1 + f_2): (X_1 + X_2) \to (Y_1 + Y_2)$

Hàm tử đồng nhất có dạng $\mathrm{Id}: Set \to Set$, $\mathrm{Id}(X) = X$, $\mathrm{Id}(f) = f: X \to Y$ với $f: X \to Y$. Với mọi tập hợp A, ta có hàm tử hằng $F_A: Set \to Set$, $F_A(X) = A$, $F_A(f) = 1_A : A \to A$ với $f: X \to Y$ và hàm tử mũ có dạng ở bảng 1.5 sau đây:

Bảng 1.5. Hàm tử và hàm tử mũ

Hàm tử (Đối tượng)	Hàm tử mũ
$(-)^A: Set \to Set$	$(-)^A(X) = X^A = \{g \mid g : A \to X\}$
$(-)^A(f): X^A \to Y^A$	$(-)^A(f)(g) = f^A(g) = f \circ g$ với $f: X \to Y, g: A \to X$

Rutten [20] phát biểu hàm tử tập của các tập con được định nghĩa chi tiết ở bảng 1.6 như sau:

Bảng 1.6. Hàm tử tập của các tập con

Hàm tử trên tập	Hàm tử của các tập con
$\mathcal{P}: Set \to Set$	$\mathcal{P}(X) = \{V \mid V \subseteq X\}$
$\mathcal{P}(f): \mathcal{P}(X) \to \mathcal{P}(Y)$	$\mathcal{P}(f)(V) = f(V)$ với $f: X \to Y, V \subseteq X$

Các hàm tử đều được tạo ra bằng cách kết hợp hai hoặc nhiều hàm tử ở trên. Hàm tử cho các máy trạng thái xác định với đầu vào từ một tập hợp S cho trước dưới dạng $\mathrm{dA}: \mathrm{Set} \to \mathrm{Set}$ sao cho $\mathrm{dA}(S) = 2 \times S^A$ được tạo ra bằng cách kết hợp hàm tử hằng F_2 và hàm tử mũ $(-)^A$ với hàm tử tích $F_\times$.

m) Đồng cấu

Nhiều tính chất cơ bản về đồng cấu của các hệ thống xử lý stream và máy trạng

thái cũng đúng cho các đồng đại số của bất kỳ loại nào. Đồng cấu là một ánh xạ bảo toàn cấu trúc đại số, tức là phép toán trong tập nguồn được bảo tồn khi chuyển sang tập đích (Bảng 1.3).

Định nghĩa 1.3 (Đồng cấu) [20]: Một đồng cấu là một ánh xạ bảo toàn cấu trúc đại số giữa hai cấu trúc đại số cùng loại. Cho hai nhóm $(A,*)$ và $(B,*)$ là một ánh xạ $f: A \to B$, là một đồng cấu nhóm nếu $f(a * b) = f(a) * f(b), \forall a, b \in A$.

Định lý 1.1 (Đồng cấu) [20]: Cho $f: (A, \alpha) \to (B, \beta)$ là một F-đồng cấu và cho $g: B \to A$ là một cấu xạ trong $\boldsymbol{C}$ là nghịch đảo của f dưới dạng $g \circ f = 1_A$ và $f \circ g = 1_B$, khi đó g là một F-Đồng cấu.

Cho $F: Set \to Set$ là một hàm tử, gọi F-Đồng cấu này $f: (A, \alpha) \to (B, \beta)$ là đơn cấu nếu với mọi $g, h: (C, \gamma) \to (A, \alpha)$, $f \circ g = f \circ h \Longrightarrow g = h$. Phát biểu này là một tính chất tổng quát về đồng cấu của các đồng đại số.

n) So sánh giữa đồng cấu, cấu xạ và hàm tử

Bảng so sánh giữa *đồng cấu*, *cấu xạ*, và *hàm tử* giúp luận án thấy rõ điểm giống và khác nhau giữa ba khái niệm quan trọng trong lý thuyết stream (Bảng 1.7).

Bảng 1.7. Bảng so sánh đồng cấu, cấu xạ, và hàm tử

Tiêu chí	Đồng cấu	Cấu xạ	Hàm tử
Cấp độ trừu tượng	Trung bình	Trừu tượng hơn	Cao
	Làm việc với cấu trúc đại số cụ thể	Trong phạm trù	Ánh xạ giữa các phạm trù
Nơi xuất hiện	Đại số trừu tượng (nhóm, vành, vector, ...)	Lý thuyết phạm trù	Lý thuyết phạm trù
Bản chất	Hàm bảo toàn cấu trúc đại số	Ánh xạ giữa các đối tượng trong phạm trù	Ánh xạ giữa phạm trù (đối tượng và cấu xạ)
Ký hiệu	$f: A \to B$ và $f(a * b) = f(a) \cdot f(b)$ với $f(*) = \cdot$	$f: A \to B$	$F: \mathcal{C} \to \mathcal{D}$
Bảo toàn cấu trúc	Cấu trúc đại số (phép toán)	Không yêu cầu cụ thể Tuỳ phạm trù	Liên kết và cấu xạ đơn vị trong phạm trù
Vai trò trong phạm trù	Là một loại cấu xạ (Trong phạm trù: Nhóm, vành, ...)	Là ánh xạ trong phạm trù	Là hàm giữa các phạm trù

o) Cơ chế lập lịch

Cơ chế lập lịch (Scheduling) trong lập trình stream là quá trình xác định *thứ tự* và *thời điểm* thực thi các stream dữ liệu trong một cơ chế đường ống xử lý dữ liệu liên tục thời gian thực. Cơ chế này đảm bảo rằng các phép biến đổi trên stream dữ liệu được thực hiện một cách tối ưu, giúp cân bằng tài nguyên hệ thống và giảm thiểu độ trễ xử lý trong quá trình vận hành theo từng phần của dữ liệu được phát biểu trong

các công trình [CT.2][CT.9].

p) Bước điều khiển

Bước điều khiển (Control step - CStep) trong lập trình stream là các giai đoạn xử lý dùng để quản lý và điều phối luồng dữ liệu, trong đó các phép toán xử lý stream được sắp đặt và kích hoạt thực thi theo pipeline. Mỗi bước điều khiển quyết định cách dữ liệu được tiếp nhận và điều phối, nhằm đảm bảo quá trình xử lý stream diễn ra liên tục trong thời gian thực đã được công bố ở các công trình [CT.2][CT.9].

1.6.2. Các phép toán stream cơ bản

Đại số stream dùng cho dữ liệu lớn *kiểu số thực, số tự nhiên, số hữu tỷ, tập mẫu tự, các bitstream* và *các mạch stream* được thể hiện qua các phép toán cơ bản ở bảng 1.8 sau đây được luận án khảo sát trong các công trình [9, 17, 20, 36-38] này:

Bảng 1.8. Các phép toán stream cơ bản

STT	Tên phép toán	Phép toán
01	Giá trị ban đầu (Head)	$\sigma(0)$
02	Đạo hàm stream (Stream derivative) (Tail)	σ'
03	Hằng stream ($r \in \mathbb{R}$), $\mathbb{R}$ là tập số thực	r
04	Biến hình thức stream số thực	X
05	Phép toán tổng hai stream số thực σ và τ	$\sigma + \tau$
06	Phép toán tích chập hai stream số thực σ và τ	$\sigma \times \tau$
07	Tổng hữu hạn của các stream số thực σ, τ, ρ, v.v.	$\sum_{n=0}^{\infty} \sigma_n$
08	Phép toán đảo ngược tích stream số thực σ, τ, ρ, v.v.	σ^{-1}
09	Phép toán liên kết hai stream số thực σ và τ	$\sigma \circ \tau$

1.6.3. Khái niệm stream

Việc xử lý dãy tuần tự các số thực hữu hạn và vô hạn được gọi chung là xử lý các stream số thực được luận án khảo sát trong các công trình [12, 17, 20, 36, 37, 39-42], $\mathbb{R}^{\omega}$ là tập các stream số thực được định nghĩa hình thức (Bảng 1.9) qua hàm (1.1) như sau:

$$\mathbb{R}^{\omega} = \{\sigma \mid \sigma : \{0, 1, 2, 3, \dots\} \to \mathbb{R}\} \qquad (1.1)$$

Trong đó:

- σ là một stream số thực có dạng $\sigma = (\sigma(0), \sigma(1), \sigma(2), \dots)$
- $\mathbb{R}^{\omega}$ là tập các stream dãy số thực vô hạn
- $\sigma(0) \in \mathbb{R}$ là phần tử đầu tiên của stream, và $\sigma' \in \mathbb{R}^{\omega}$ là đạo hàm của stream, lấy phần còn lại của stream sau khi loại bỏ phần tử đầu tiên.

Bảng 1.9. Phân tích giá trị ban đầu và đuôi của stream

Đuôi (Tail)	Đầu (Head)	Điều kiện
$\sigma' = (\sigma(1), \sigma(2), \dots)$	$\sigma(0)$	$0 \le i \le \omega$

Zakowski và cộng sự [43] xét loại dữ liệu stream đồng quy nạp (các stream cũng có thể là hữu hạn) và phát biểu một số mẫu stream điển hình trong nghiên cứu stream

đồng quy nạp có tính chất khái quát hóa tham số. Các mẫu stream này có các dạng thức như sau:

- Dạng stream hữu hạn: 01ϵ hoặc $\tau 0\tau\tau 1\epsilon$
- Dạng stream tăng vô hạn: $012 \ldots n(n+1)$ ... hoặc $0\tau 1\tau 2 \ldots n\tau(n+1)$ ...
- Dạng stream xen kẽ vô hạn: 01010101 ...
- Dạng stream phân kỳ ngầm (phân kỳ tiềm ẩn): $\tau\tau\tau\tau\tau\tau\tau$ ...

Các dạng đặc điểm stream ở trên có thể được sử dụng để đại diện cho dấu vết của một hệ thống chuyển đổi trạng thái. Đối tượng stream như vậy là một dãy tuần tự các sự kiện bên trong tiềm ẩn vô hạn, τ, các sự kiện bên ngoài $\beta(n)$, và kết thúc nếu hữu hạn bằng bộ đánh dấu ϵ. Để đơn giản hóa, nhóm tác giả này đã giả định rằng các sự kiện có thể nhìn thấy mang theo một số tự nhiên. $\beta(n)$ là cách ghi đầy đủ để biểu diễn một sự kiện có thể quan sát được với giá trị n trong stream, nhưng trong các mẫu stream được minh họa ở trên, ký hiệu này bị lược bỏ để đơn giản hóa cách trình bày.

Nhóm tác giả ký hiệu $\beta(n)$ được dùng để biểu diễn sự kiện ngoại vi mang giá trị là một số tự nhiên n hoặc còn được gọi là sự kiện có thể quan sát được trong một stream của một hệ thống chuyển trạng thái. Đây là một phần của chuỗi các sự kiện trong một stream, mà stream đó có thể bao gồm: τ là các sự kiện nội bộ không thể quan sát từ bên ngoài; $\beta(n)$ là các sự kiện ngoại vi có thể quan sát được và mang giá trị cụ thể; ϵ là đánh dấu kết thúc của stream (nếu stream hữu hạn). Xem qua hai ví dụ minh họa như sau:

- **Ví dụ 1.2:** Stream $s_0 = 01\epsilon$ tương đương với $\beta(0)\beta(1)\epsilon$ là chuỗi gồm hai sự kiện ngoại vi mang giá trị 0 và 1, và sau đó kết thúc.
- **Ví dụ 1.3:** Stream $s_1 = \tau 0\tau 1\varepsilon$ tương đương với $\tau\beta(0)\tau\beta(1)\varepsilon$ là biểu diễn xen kẽ các sự kiện nội bộ và ngoại vi, kết thúc bằng ε.

1.6.4. Đồng quy nạp dùng cho stream

Đồng quy nạp là một kỹ thuật chứng minh được sử dụng để xác lập tính đồng nhất của hai stream hoặc để chứng minh một phần tử thuộc về một tập hợp vô hạn. Trái ngược với quy nạp, đồng quy nạp không chứng minh tính đúng đắn bằng cách đi từ trường hợp cơ sở đến trường hợp tổng quát, mà thay vào đó sử dụng các đặc điểm lặp lại hoặc các định nghĩa vòng lặp để chứng minh rằng hai đối tượng có cấu trúc tương tự nhau.

Với tập $\mathbb{R}^\omega$ của tất cả các stream số thực được mô tả ở *tiểu mục 1.6.3*, cho thấy $\mathbb{R}^\omega$ đáp ứng nguyên lý chứng minh đồng quy nạp. $\mathbb{R}^\omega$ chỉ ra tính chất chung của một máy trạng thái stream cuối được gọi là đồng đại số.

Tập $\mathbb{R}^\omega$ được định nghĩa hình thức ở (1.1). Trong đó, $\sigma(0)$ là giá trị ban đầu của σ, đạo hàm của σ có dạng $\sigma'(n) = \sigma(n+1)$ là các giá trị còn lại của σ. Tập $\mathbb{R}^\omega$

được xử lý như những thực thể toán học và tùy thuộc vào ngữ cảnh khi đề cập đến các phần tử đơn lẻ do $\mathbb{R}^\omega$ tạo ra, ký hiệu hai stream σ và τ có dạng bảng 1.10.

Bảng 1.10. Phân tích hai stream σ và τ số thực $\mathbb{R}$

Streams σ và τ số thực	Đạo hàm của streams σ và τ số thực
$\sigma = (\sigma(0), \sigma(1), \sigma(2), \dots) = (s_0, s_1, s_2, \dots)$	$\sigma' = (s_1, s_2, \dots)$
$\tau = (\tau(0), \tau(1), \tau(2), \dots) = (t_0, t_1, t_2, \dots)$	$\tau' = (t_1, t_2, \dots)$

$\forall n \geq 0$, ta có số thực $\sigma(n) = s_n$ được gọi là phần tử thứ n của stream σ. Số thực này cũng được biểu đạt theo đạo hàm bậc cao của stream σ, $\forall k \geq 0$, đạo hàm bậc cao của stream σ có dạng bảng 1.11.

Bảng 1.11. Phân tích đạo hàm bậc cao của stream σ số thực

Stream σ số thực	Đạo hàm bậc cao của stream σ số thực	Điều kiện
$\sigma^{(0)} = \sigma$	$\sigma^{(k+1)} = \left(\sigma^{(k)}\right)'$	$k \geq 0$

Ngoài ra, phần tử thứ n của stream σ và τ được cho tương ứng bởi $\sigma(n) = \sigma^{(n)}(0)$ và $\tau(n) = \tau^{(n)}(0)$. Để có được hai stream σ và τ bằng nhau, phải có điều kiện cần và đủ để chứng minh $\forall n \geq 0$, suy ra $\sigma(n) = \tau(n)$.

1.6.5. Mô phỏng hai chiều

Định nghĩa 1.4 (Mô phỏng hai chiều) [17]: Một quan hệ $R \subseteq \mathbb{R}^\omega \times \mathbb{R}^\omega$ được gọi là một mô phỏng hai chiều (Bisimulation) trên $\mathbb{R}^\omega$ nếu với mọi cặp $(\sigma, \tau) \in \mathbb{R}^\omega \times \mathbb{R}^\omega$, nếu $\sigma\ R\ \tau$ thì: $\begin{cases} \sigma(0) = \tau(0) \\ \sigma'\ R\ \tau' \end{cases}$, trong đó $\sigma(0)$ và $\tau(0)$ là phần đầu của σ và τ, còn σ' và τ' là phần đuôi của σ và τ, tức là stream còn lại sau phần tử đầu tiên. Khi đó, σ và τ là một mô phỏng hai chiều, ký hiệu là: $\langle \sigma, \tau \rangle \in R$ hoặc $\sigma \sim \tau$.

Định lý 1.2 (Đồng quy nạp) [17]: Nếu hai stream σ và τ là mô phỏng hai chiều ($\sigma \sim \tau$), thì suy ra rằng $\sigma(n) = \tau(n)$, $\forall n \geq 0$, và do đó $\sigma = \tau$. Nói cách khác, $\forall \sigma, \tau \in \mathbb{R}^\omega$, ta có: $\sigma \sim \tau \Rightarrow \sigma = \tau$.

1.6.6. Phương trình vi phân hành vi

Các stream và các phép toán xử lý stream được đặc tả thông qua các phương trình vi phân hành vi, xác định các đạo hàm và giá trị ban đầu của chúng. Rutten [17, 20, 37], nhóm tác giả [40] đưa ra minh họa cơ bản về sự tồn tại của một stream σ thỏa mãn phương trình vi phân hành vi ở bảng 1.12 (Stt: 1) như sau:

Bảng 1.12. Phương trình vi phân hành vi của một stream σ/stream hằng

Stt	Phương trình vi phân hành vi (Tail)	Giá trị ban đầu (Head)	Tên/Điều kiện
1	$\sigma' = \sigma$	$\sigma(0) = 1$	Stream σ
2	$[r]' = [0]$	$[r](0) = r$	Stream hằng
3	$\mathrm{X}' = [1]$	$\mathrm{X}(0) = 0$	Biến hình thức
4	$\left(\sigma^{(n)}\right)' = \sigma^{(n+1)}$	$\sigma(0)$	$n \geq 0$

Xét stream $\sigma = (1, 1, 1, \dots)$ với $\sigma' = \sigma$ và $\sigma(0) = 1$, Rutten [20] chứng minh rằng đây là nghiệm duy nhất của phương trình vi phân hành vi. Bằng cách tiếp cận đồng quy nạp trên tập $\mathbb{R}^\omega$ (Tập các dãy số thực vô hạn) và sử dụng tính chất cuối,

Rutten định nghĩa một máy trạng thái hình thức $(S, \langle o, t \rangle)$ với:

- Tập các trạng thái $S = \{s\}$,
- Hàm chuyển đổi trạng thái $t: S \to S$ sao cho $t(s) = s$,
- Hàm quan sát $o: S \to \mathbb{R}$ cho $o(s) = 1$.

Cấu trúc máy trạng thái này mô hình hóa đúng hành vi của stream σ, theo đúng phương trình vi phân hành vi $\sigma' = \sigma$, với điều kiện khởi đầu $\sigma(0) = 1$.

1.6.7. Stream hằng

Rutten và cộng sự trong các công trình [12, 16, 17, 20, 36, 37, 44] gọi số thực trong $\mathbb{R}^\omega$ là một stream hằng. Với $r \in \mathbb{R}$, gọi $[r]$ là stream duy nhất thỏa mãn phương trình vi phân hành vi ở bảng 1.12 (Stt: 2).

Để dễ dàng quan sát và sử dụng stream hằng so với việc sử dụng phương trình vi phân hành vi, Rutten đã đưa ra định nghĩa rõ ràng hơn về stream hằng (1.2) sau:

$$[r] = (r, 0, 0, 0, 0, \dots) \quad (1.2)$$

Việc đưa các số thực $\mathbb{R}$ vào tập hợp các stream thông qua phép toán $[]: \mathbb{R} \to \mathbb{R}^\omega$, lúc này $[r]$ với số thực r của nó biểu diễn. Do đó, thường viết r thay cho $[r]$. Phương trình sau đây định nghĩa thêm một hằng gọi là hằng X ở bảng 1.12 (Stt: 3). Hằng X đóng vai trò quan trọng trong tính toán stream được gọi là một biến hình thức (1.3) như sau:

$$X = (0, 1, 0, 0, 0, \dots) \quad (1.3)$$

1.6.8. Đại số stream

Khái niệm cơ bản về stream và các phép toán stream cơ bản, Rutten và cộng sự trong các công trình [10, 20, 45] đề xuất lý thuyết stream (đại số stream và đồng đại số stream). Đại số stream gồm các phép toán xử lý stream dùng cho stream, gồm: cộng, tích trộn, tích, đảo ngược, thanh ghi, các phép toán này xử lý tập dữ liệu kiểu $\mathbb{R}$. Stream σ được Rutten định nghĩa dưới dạng một hàm stream (Stream function) $\mathbb{R}$ như sau (1.4):

$$\sigma: \mathbb{N} \to \mathbb{R} \quad (1.4)$$

Trong đó:

- σ: là stream có dạng $\sigma = (\sigma(0), \sigma(1), \sigma(2), \dots)$.
- $\mathbb{N} = \{0, 1, 2, \dots\}$: là tập hợp các số tự nhiên.
- $\mathbb{R}$: là tập hợp các số thực.

Với hầu hết định nghĩa hình thức về stream σ, Rutten [20] sử dụng khái niệm đạo hàm stream như bảng 1.12 (Stt: 4). Trong đó, $\sigma(0)$ là giá trị ban đầu của stream σ và σ' là đạo hàm của stream σ bằng phương trình vi phân sau $(\sigma^n)' = \sigma^{n+1}$.

1.6.9. Đồng đại số stream

Rutten đề xuất đồng đại số stream trong các công trình số [17, 20] dùng cho $\mathbb{R}^\omega$, xử lý dãy tuần tự các số thực vô hạn được gọi là các stream. Một tính toán stream được Rutten phát triển theo hai cách như sau:

- Trong giải tích: Đề cập đến vi phân stream.
- Trong đại số: Đề cập đến tính toán với các phép toán và thiết lập đồng nhất.

Tập $\mathbb{R}^\omega$ mang một cấu trúc đồng đại số cuối được mô tả bởi cặp phép toán stream $\langle O, \mathrm{T}\rangle$ ở bảng 1.13. Chúng gán cho một stream $\sigma = (\sigma(0), \sigma(1), \sigma(2), \dots)$, với $\sigma(0) \in \mathbb{R}$ được gọi là đầu và đạo hàm $\sigma' \in \mathbb{R}^\omega$ được gọi là đuôi ở bảng 1.14.

Bảng 1.13. Một cấu trúc đồng đại số stream dùng cho $\mathbb{R}$

Cặp phép toán $\langle O, \mathrm{T}\rangle$	Phần tử stream dạng $\mathbb{R}^\omega$
$\mathbb{R}^\omega \xrightarrow[\langle O,\mathrm{T}\rangle]{} \mathbb{R} \times \mathbb{R}^\omega$	$\sigma \longmapsto \langle \sigma(0), \sigma' \rangle$

Bảng 1.14. Phương trình vi phân hành vi stream dùng cho $\mathbb{R}$

Giá trị cuối $\langle \mathrm{T} \rangle$	Giá trị ban đầu $\langle O \rangle$
$\mathbb{R}^\omega \to \mathbb{R}^\omega$	$\mathbb{R}^\omega \to \mathbb{R}$
$\sigma' = (\sigma(1), \sigma(2), \sigma(3), \dots)$	$\sigma(0)$

1.6.10. Ngôn ngữ RTL

RTL [46] là biểu diễn trung gian gần với hợp ngữ, dùng để mô tả, thiết kế và kiểm chứng các hệ thống, bao gồm tính toán BDL dựa trên phép tính stream, được luận án sử dụng và phân tích chi tiết ở *Tiểu mục 3.2.1* của *Chương 3*.

1.6.11. Không gian Chu

Định nghĩa 1.5 [47, 48]: Một không gian Chu trên tập giá trị K (chẳng hạn $K = \{0,1\}$) là một bộ ba (X, r, A), trong đó:

- X là tập hợp các trạng thái. A là tập hợp các sự kiện.
- $r: X \times A \to K$ là một hàm quan hệ, ánh xạ mỗi cặp $(s, e) \in X \times A$ tới một giá trị trong K.

1.6.12. Một số quan hệ giữa trạng thái và sự kiện trong không gian Chu

Một số mối quan hệ nền tảng giữa các sự kiện, bao gồm: Quan hệ độc lập (∇), quan hệ trước sau ($\prec$), quan hệ xung đột ($\#$), và quan hệ khả năng kích hoạt phân tách (**den**) được định nghĩa trong các nghiên cứu số [49, 50] và được luận án sử dụng và phân tích chi tiết ở *Tiểu mục 4.3* của *Chương 4*.

1.7. Phân tích các nghiên cứu liên quan

Các nghiên cứu về đại số stream của Rutten và một số nhóm tác giả được luận án khảo sát và phân tích. Các thành phần của đại số stream dùng cho tập stream là dãy các số thực vô hạn và khái niệm đạo hàm stream, các chứng minh đồng quy nạp, và một số định nghĩa được hình thức hóa.

Rutten [16] xây dựng các sơ đồ luồng tín hiệu và chứng minh các kết quả đã biết về mạch stream bằng cách sử dụng đại số stream. Các sơ đồ này được tạo thành

từ những phép toán cơ bản: bộ nhân, bộ sao chép, bộ cộng và bộ thanh ghi. Đặc biệt, Rutten đề xuất các mạch stream đồng thời kết hợp các mạch stream này lại với nhau, từ đó hình thành các định nghĩa và định lý liên quan đến mạch lặp phản hồi. Một hệ quả quan trọng được Rutten phát biểu: Một stream $\rho \in \mathbb{R}^{\omega}$ là hữu tỷ nếu và chỉ nếu nó có thể được phát sinh bởi một mạch stream hữu hạn.

Rutten [15] định nghĩa, phân tích và liên kết thống nhất bốn cấu trúc đại số khác nhau trên tập hợp của các stream bit làm nền tảng cho chúng được mô tả dưới dạng các mạch số. Với một trong những cấu trúc đại số này, có các số 2 toán hạng, Rutten đã đặc tả một lớp các mạch số tuyến tính dưới dạng các stream hữu tỉ.

Nghiên cứu [13] trình bày cách lập luận về stream là dãy vô hạn thuộc miền của một kiểu dữ liệu đồng quy nạp, với các phép toán thực hiện qua chương trình đồng đệ quy và chứng minh bằng đồng quy nạp. Nghiên cứu cũng đề xuất phương trình stream (Stream equation) như một giải pháp thay thế, đảm bảo tồn tại nghiệm.

Milius [14] nghiên cứu tính đúng đắn và đầy đủ của các mạch stream hữu hạn, xem chúng như biểu diễn đồ họa trực quan cho các hàm stream được tính bằng hệ thống tuyến tính hữu hạn chiều. Milius đưa ra một biểu thức cho phép suy luận sự tương đương ngữ nghĩa giữa các mạch stream đóng hữu hạn, dựa trên đặc tả cú pháp đồng đại số hữu hạn cục bộ cuối.

Kupke và cộng sự [12] đề xuất một định dạng cú pháp cho các phương trình vi phân stream (Stream differential equations), trong đó mọi hệ phương trình tuân theo định dạng này đều có nghiệm duy nhất. Định dạng này cho phép xác định các hàm stream một cách rõ ràng. Nhóm tác giả cũng thảo luận về phép tính stream phi chuẩn (Non-standard stream calculus), sử dụng các phép toán khác với phép toán đầu và đuôi để xác định và lập luận về stream và hàm stream.

Stream là các đối tượng vô hạn với lý thuyết phong phú trong toán học và khoa học máy tính [10]. Phương trình vi phân stream là một phương pháp đồng quy nạp để định nghĩa stream và các phép toán trên chúng. Lý thuyết này đã được phát triển mạnh, với nhiều định dạng, phương pháp giải và phân loại các lớp stream.

Nghiên cứu hệ thống phát triển kỹ thuật mô phỏng hai chiều cho đồng đại số, mở rộng chứng minh mô phỏng cho nhiều hệ thống dựa trên trạng thái: Hệ thống chuyển đổi nhãn, stream và máy trạng thái. Đặc điểm là khả năng lập luận tính đúng đắn của các cải tiến. Nghiên cứu tập trung vào các phép toán như tách, phân vùng, chiếu, trộn stream dữ liệu, vốn quan trọng trong lập trình và ngôn ngữ xử lý stream. Các phép toán này được định nghĩa và chứng minh thuộc tính bằng phép tính stream, mạch stream, phương trình vi phân và nguyên lý đồng quy nạp [9].

Nghiên cứu sự bất biến của một số lớp mẫu tốt của các stream [11], cụ thể là

các stream hữu tỉ và các stream đại số dưới dạng phép toán tách và trộn. Nghiên cứu chỉ ra các mạch stream được mở rộng với các cổng để tách và hợp nhất dị hợp là đủ biểu cảm để nhận ra một số stream đại số phi hữu tỉ, do đó vượt ra ngoài các mạch stream thông thường.

Phân tích và tính toán stream là một chủ đề ngày càng thu hút trong khoa học máy tính. Đại số stream của dãy tuần tự $(1 \perp -)$ gồm các dãy vô hạn $\{0, 1, \perp\}$ đã được nghiên cứu và ứng dụng trong tính toán số thực, với khoảng đơn vị $\mathbb{I}$ được nhúng vào tập stream $\sum_{\perp,1}^{\omega}$ của dãy $\{0, 1, \perp\}$. Một hàm thực trên $\mathbb{I}$ có thể được biểu diễn như chương trình với đầu vào và đầu ra là các dãy tuần tự [18]. Việc hình thức hóa tính toán trên stream trong ngôn ngữ lập trình cũng đã được nghiên cứu trong các công trình [19, 21].

Các nghiên cứu gần đây về lập lịch stream đề xuất nhiều hướng tiếp cận. Mô hình Dataflow của Akidau [51] là nền tảng cho Google Dataflow và Apache Flink, hỗ trợ xử lý stream mở rộng với ngữ nghĩa theo thời gian sự kiện, watermark và các cửa sổ (tumbling, sliding, session), nhưng thiếu lập lịch pipeline và mô hình hóa phụ thuộc tác vụ cho bài toán BDL phức tạp. Mao [52] và Zhou [53] áp dụng học tăng cường sâu cho quản lý tài nguyên động trong stream và cloud, tuy có khả năng thích ứng và học liên tục cao nhưng khó tích hợp vào pipeline do phức tạp huấn luyện, tiêu tốn tài nguyên và khó mô hình hóa phụ thuộc pipeline. Tương tự, học chủ động của Cacciarelli [54] hỗ trợ thích ứng trực tuyến qua chọn mẫu thông tin nhưng chủ yếu cải thiện độ chính xác mô hình hơn là tối ưu pipeline.

Xét các phương pháp lập lịch tối ưu hóa, Tantalaki và công sự [55] đề xuất mô hình quy hoạch tuyến tính kết hợp phương trình Diophantine cho lập lịch định kỳ trong hệ thống stream phân tán. Shukla [56] giới thiệu lập lịch điều khiển theo mô hình, dự đoán tài nguyên và độ trễ để phân bổ tác vụ thích ứng trên Apache Storm. Nghiên cứu này phát triển mô hình lập lịch pipeline thống nhất dựa trên quy hoạch tuyến tính, đồng thời tối ưu phụ thuộc tác vụ, bước điều khiển, tái sử dụng tài nguyên và tránh xung đột trong xử lý stream thời gian thực.

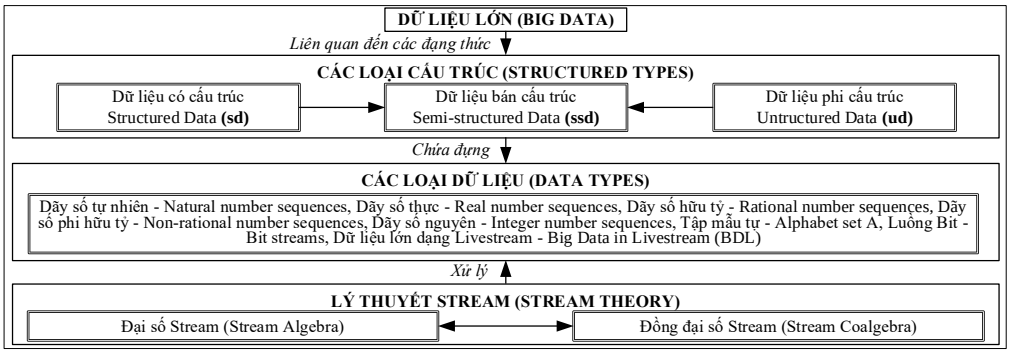
1.8. Phân tích và so sánh lý thuyết stream

Qua việc phân tích những đặc điểm của các công trình nghiên cứu liên quan lý thuyết stream. Trong phần này, luận án thực hiện rút trích ra các tiêu chí đặc trưng của mỗi công trình nghiên cứu liên quan đã đóng góp vào lý thuyết stream, qua đó lập bảng phân tích và so sánh [CT.8].

1.8.1. Phân tích lý thuyết stream

Luận án tiến hành phân tích các nghiên cứu liên quan dựa trên hai tiêu chí đó là áp dụng lý thuyết stream và các loại cấu trúc dữ liệu (Hình 1.9). *Tiêu chí 1*: Áp dụng

lý thuyết stream (Đại số stream và đồng đại số stream) vào xử lý dữ liệu. *Tiêu chí 2*: Việc áp dụng vào các loại cấu trúc dữ liệu (Ký hiệu *sd* là dữ liệu có cấu trúc, *ssd* là dữ liệu bán cấu trúc, và *ud* là dữ liệu phi cấu trúc). Bảng 1.15 và 1.16 thể hiện phân tích hai tiêu chí này như sau [CT.8].



Hình 1.9. Các loại cấu trúc dữ liệu dùng cho lý thuyết stream

Bảng 1.15. Phân tích đại số stream qua việc áp dụng và loại cấu trúc

Công trình	Áp dụng đại số stream (Applying stream algebra)	Loại cấu trúc
Kovach và cộng sự, 2023 [57]	Tính toán biểu thức rút gọn bằng các stream lồng nhau, yêu cầu thuộc tính thống nhất về thứ tự cho mọi stream đầu vào.	*ud*
Helala và cộng sự, 2022 [33]	Áp dụng cho hệ thống thị giác máy tính quy mô lớn và áp dụng cơ chế đường ống (pipeline) xử lý của nó.	*ud*
Michele và cộng sự, 2021 [40]	Để liên kết đa thức, phương trình vi phân và các stream.	*ud*
Mauricio và cộng sự, 2018 [58]	Các phép toán cơ bản trên đại số stream áp dụng cho dãy số thực và dãy số tự nhiên.	*ud*
Helle và cộng sự, 2017 [10]	Phương trình vi phân stream (đồng quy nạp) đặc tả stream và các phép toán trên stream.	*ud*
Giorgio và cộng sự, 2015 [59]	Để mô hình hóa cú pháp, ngữ nghĩa của tiến trình ngẫu nhiên với trạng thái liên tục của hàm nội sinh trong không gian đo lường.	*ud* *ssd* *sd*
Filippo và cộng sự, 2014 [60]	Để giúp tối thiểu hóa máy trạng thái hữu hạn, chứng minh tính đúng đắn và mở ra cách tiếp cận tổng quát hơn.	*sd* *ssd* *ud*
Niqui và cộng sự, 2013 [11]	Phép tính và mạch stream định nghĩa, chứng minh tính chất các phép toán stream: splitting, partitioning, projecting, merging.	*sd* *ssd* *ud*
Kupke và cộng sự, 2011 [12]	Phân tích phép tính stream phi chuẩn, sử dụng phép toán head và tail để định nghĩa và lập luận về stream và hàm stream.	*sd* *ssd* *ud*
Milad và cộng sự, 2010 [61]	Áp dụng đại số stream nghiên cứu các phép toán như partitioning, projecting, merging và splitting trong lập trình và ngôn ngữ xử lý stream dữ liệu.	*sd* *ssd* *ud*

Vinh và cộng sự, 2008 [62]	Áp dụng đại số stream nghiên cứu lập trình và lập lịch stream dữ liệu cho cấu hình đầu vào.	*ud*
Rutten, 2005 [63]	Áp dụng phép tính stream đồng quy nạp cho sơ đồ luồng tín hiệu và xử lý hình thức cho chúng.	*ud*
Rutten, 2005 [15]	Áp dụng bốn cấu trúc đại số trên tập bitstream làm nền tảng mô tả dưới dạng mạch kỹ thuật số.	*sd* *ssd*
Rutten, 2004 [16]	Áp dụng đại số stream gồm các phép toán bộ nhân, bộ sao chép, bộ cộng và bộ thanh ghi để xây dựng sơ đồ luồng tín hiệu.	*sd* *ud*
Rutten, 2001 [17]	Áp dụng tính toán trên stream gồm phương trình vi phân, phân số liên tục và các bài toán rời rạc, tổ hợp. Tác giả phát triển phép tính stream đồng quy nạp cho dãy số thực vô hạn.	*sd* *ssd* *ud*

Bảng 1.16. Phân tích đồng đại số stream qua việc áp dụng và loại cấu trúc

Công trình	**Áp dụng đồng đại số stream (Applying stream coalgebra)**	**Loại cấu trúc**
Michele và cộng sự, 2024 [39]	Liên kết đa thức, phương trình vi phân và các stream trên trường $\mathbb{K}$ để xây dựng lớp (F, G) trên stream.	*ud*
Florian 2022 [64]	Cho ngữ nghĩa của máy trạng thái không xác định và định danh.	*sd* *ssd*
Rutten, 2019 [20]	Để nghiên cứu hành vi của hệ thống động dựa trên trạng thái.	*ud*
Henning, 2019 [65]	So sánh bốn phép toán tích trên ngôn ngữ có trọng số: convolution, shuffle, infiltration và Hadamard, và phát hiện tập ngôn ngữ có trọng số là một đồng đại số stream cuối.	*sd* *ssd* *ud*
Filippo và cộng sự, 2017 [66]	Ứng dụng ngữ nghĩa toán học cho các phép toán và phép tính trên stream số thực vô hạn.	*sd* *ssd* *ud*
Rot và cộng sự, 2016 [67]	Sự tương đương và đa dạng của ngôn ngữ hình thức được kiểm tra đồng quy nạp qua mô phỏng hai chiều trên máy trạng thái tách biệt.	*sd* *ssd* *ud*
Joost và cộng sự, 2015 [41]	Để biểu đạt và đặc trưng hóa ngôn ngữ phi ngữ cảnh qua hàm stream và hệ phương trình vi phân hành vi.	*sd* *ssd* *ud*
Helle và cộng sự, 2014 [68]	Nghiên cứu dãy tuần tự k từ quan điểm đồng đại số stream, xây dựng trên tập stream của nửa vòng S như một đồng đại số cuối. Họ đặc trưng dãy tuần tự k qua máy trạng thái có trọng số hữu hạn, hệ phương trình vi phân và chuỗi mũ.	*ud*
Rot và cộng sự, 2013 [69]	Mô phỏng hai chiều làm đơn giản hóa đồng đại số stream và mở rộng phân tích hành vi các cấu trúc đồng đại số, nâng cao phương pháp chứng minh.	*ud*
Venanzio, 2011 [70]	Trong lập trình hàm và lý thuyết loại, với lý thuyết cơ bản triển khai qua các loại đồng quy nạp. Nghiên cứu ứng dụng phương pháp đồng đệ quy cho đệ quy tổng quát, chuỗi mũ hình thức và lập bảng hàm trên dữ liệu quy nạp.	*sd* *ssd* *ud*
Silva và cộng sự, 2010 [71]	Nghiên cứu tập hợp các cây nhị phân vô hạn T^A với nodes gán nhãn nửa vành A từ góc nhìn đồng đại số.	*sd* *ssd* *ud*

Marcello và cộng sự, 2009 [72]	Đồng đại số đa thức mô tả các hệ thống động, như máy trạng thái xác định và hệ thống chuyển trạng thái gán nhãn, thông qua các hàm tử đa thức.	*ud*
Rutten, 2008 [73]	Nghiên cứu stream hữu tỷ trên một trường dưới góc nhìn đồng đại số và khai thác tính hữu hạn của tập các stream.	*ud*
Rutten, 2006 [74]	Mô tả đạo hàm stream và khái quát hóa đạo hàm của các biểu thức hữu tỷ cho hàm stream với mẫu tự đầu vào và đầu ra tùy ý.	*ud*
Rutten, 2005 [36]	Phát triển cấu trúc đồng quy nạp cuối trên tập stream để xây dựng phép tính đồng quy nạp dưới dạng dãy số thực vô hạn.	*ud*

1.8.2. Phân tích và so sánh lý thuyết stream trên các tiêu chí

Bảng 1.15 và bảng 1.17, luận án nhận thấy rằng việc phân tích lý thuyết stream mang lại những lợi ích trong việc xử lý dữ liệu lớn, giúp xác định các đặc trưng của lý thuyết stream đã đóng góp. Các nghiên cứu liên quan đã giải quyết được các loại dữ liệu đa dạng, tuy nhiên vẫn chưa chú trọng đến stream dạng BDL trong các hệ thống tính toán. Vì vậy, luận án xây dựng bảng 1.17 nhằm phân tích và so sánh các công trình nghiên cứu trước đây so với các công trình nghiên cứu đã công bố của luận án dựa trên hai tiêu chí đặc trưng đã được công bố ở công trình [CT.8] như sau:

- *Tiêu chí 1*: Cấu trúc, tính chất, và phép toán
- *Tiêu chí 2:* Phạm vi và áp dụng

Bảng 1.17. Phân tích và so sánh nghiên cứu liên quan so với nghiên cứu của luận án

Các công trình nghiên cứu liên quan			**Các công trình nghiên cứu của luận án**		
Công trình	**Phạm vi và áp dụng**	**Cấu trúc, tính chất, và phép toán**	**Công trình**	**Cấu trúc, tính chất, và phép toán**	**Phạm vi và áp dụng**
Mauricio và cộng sự, 2018 [58]	Biểu thức đại số minh họa quá trình xử lý tín hiệu bằng cách kết hợp các phép toán một ngôi và hai ngôi, thể hiện các bước: thu thập, tạo độ trễ, chuẩn hóa, khuếch đại, đảo ngược và tổng hợp để tạo đầu ra.	Sequential composition, Parallel composition, Transformation, Synchronized parallel composition, Restriction, Reverse, Delays, Extract, Sum	Dang Van Pham và cộng sự, 2024 [CT.3]	Xây dựng cấu trúc đại số monoid, các tính chất monoid biến đổi stream, các máy trạng thái, các bộ tổ hợp (Combinators) máy trạng thái dùng cho stream dạng BDL.	Biểu thức đại số mô tả quá trình đặc tả cơ chế hoạt động của stream dạng BDL trong các hệ thống tính toán.
Michele, 2021 [40]	Các phép toán này là minh họa của tích trên các stream thể hiện việc dùng đại số và đồng đại số để tìm biểu thức đóng cho hàm sinh đại số và nghiệm của phương trình vi phân.	Sum, Product, Shuffle, Hadamard	Dang Van Pham và cộng sự, 2023 [CT.1] [CT.2]	Xây dựng các phép toán xử lý stream dạng BDL: - Dropping operator (Phép loại bỏ), - Taking operator (Phép toán lấy),	Mười phép toán stream BDL là các biểu thức đại số. ▪ **Phép toán một ngôi:** - **Phép toán dropping** loại bỏ frame thứ *k* của từng khối
Niqui và	Các phép toán này	Splitting,			

Rutten, 2013 [11]	và các tính chất của nó dùng để phân tách, chiếu và hợp nhất các stream. Trong đó phép toán take và zip dùng để trộn và tách stream mà vẫn bảo toàn được lớp stream hữu tỉ và stream đại số có cấu trúc và mẫu rõ ràng.	Partitioning, Projecting, Merging, Take, Zip		- Zipping operator (Phép toán nén), - Splitting operator (Phép toán tách), - Reverse operator (Phép toán đảo ngược), - Merging operator (Phép toán trộn), - Convolution product operator (Phép toán tích chập), - Copying operator (Phép toán sao chép), - Registering operator (Phép toán thanh ghi), - Assignment operator (Phép toán gán)	đầu vào kích thước l trong stream. - **Phép toán taking** lấy từng frame trong stream. - **Phép toán zipping** là nén từng frame trong stream. - **Phép toán splitting** là tách từng frame trong stream. - **Phép toán reverse** được tổ hợp từ *phép toán zipping* và *taking* để đảo ngược các frame trong một stream. - **Phép toán copying** dùng để sao chép một stream thành hai stream. - **Phép toán registering** được dùng để lưu kết quả trung gian của stream vì có stream đơn vị độ trễ. ▪ **Phép toán hai ngôi:** - **Phép toán merging** được sử dụng để trộn hai stream thành một stream
Rutten, 2004 [16]	Các phép toán này cho phép tính tổng, tích chập và đảo ngược stream $\mathbb{R}^{\omega}$.	Sum, Convolution product, Inverse.			
Rutten, 2019 [20]	Các phép toán cộng, tích và trừ trên tập số nguyên $\mathbb{Z}$, phép toán loại bỏ, cộng cục bộ, đảo ngược stream $\mathbb{N}$ và tách đạo hàm stream.	Addition, Multiplication, Minus, Drop, Partial sum, Inverse, Splitting stream Derivatives			
Henning và cộng sự, 2019 [65]	Các phép toán này lần lượt nối chuỗi, trộn xen không đồng bộ, trộn xen đồng bộ, và xen kẽ hoàn toàn đồng bộ giữa các hệ thống stream tạo thành một vành có một tính toán hành vi.	Convolution, Shuffle, Infiltration, Hadamard, Inverse, Shuffle inverse.			
Kupke và cộng sự, 2012 [42]	Biểu diễn tập hợp stream mẫu tự A mang một cấu trúc đồng đại số stream cuối dạng: $(A^{\omega}, \langle \text{head}, (\text{even}, \text{odd}) \rangle$ *Trong đó:* Head(0)=$\sigma(0) \in A$ và $\text{even}(\sigma) = \sigma(0), \sigma(2)$ $\text{odd}(\sigma) = \sigma(1), \sigma(3)$	Head, Tail, Even, Odd.			
Silva và cộng sự, 2010 [71]	Các phép toán này dùng để mô hình hóa các cây nhị phân dưới dạng chuỗi mũ hình thức tạo thành một đồng đại số cuối.	Head, Tail, sum, Convolution product, Star, Inverse.			
Rutten, 2008 [73]	Các phép toán này biểu diễn đầu, đuôi, tổng, tích chập, và	Head, Tail, Sum,			

	đảo ngược tích của các stream hữu tỉ.	Convolution product, Multiplicative inverse.			nhưng vẫn duy trì bản chất của stream.
Rutten, 2006 [74]	Các phép toán này biểu diễn đầu, đuôi, tổng, hiệu, tích, và đảo ngược tích các bitstream.	Head, Tail, Sum, Minus, Product, Inverse.			- **Phép toán convolution product** được dùng để triển khai trộn các frame trong hai stream.
Rutten, 2005 [36]	Các phép toán này biểu diễn đầu, đuôi, tích chập, tích trộn và đảo ngược của các stream số thực mang một cấu trúc đồng đại số cuối.	Head, Tail, Convolution product, Shuffle product, Inverse			- **Phép toán assignment** được dùng để gán stream với stream.

1.8.3. Vai trò của mười phép toán xử lý stream dạng BDL

Phân tích và so sánh trong các bảng 1.15–1.17 cho thấy rằng các nghiên cứu trước đây chủ yếu sử dụng phép toán rời rạc để xử lý các dãy tuần tự hữu hạn, dựa trên các kiểu dữ liệu như $\mathbb{N}$, $\mathbb{R}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}^{\omega}$, mẫu tự A, bitstream và mạch stream. Trong khi đó, bảng 1.17 trình bày hệ thống mười phép toán được luận án phát triển riêng cho stream dạng BDL, nhằm xử lý cả dữ liệu hữu hạn và vô hạn trong thời gian thực. Cách tiếp cận này cho phép xây dựng các phép toán stream một ngôi và hai ngôi, hướng đến việc hình thức hóa cơ chế hoạt động của các stream phi cấu trúc được định nghĩa chi tiết tại *Tiểu mục 2.3.3, trong đó Tiểu mục 2.3.3.11 trình bày những lợi ích cụ thể của các phép toán xử lý stream dạng BDL trong các CS* (*Chương 2*).

Những phân tích lý thuyết nền tảng ở *tiểu mục 1.6*, các nghiên cứu liên quan ở *tiểu mục 1.7*, và phân tích và so sánh lý thuyết stream ở *tiểu mục 1.8*, từ đó luận án nhận thấy rằng những thế mạnh và hạn chế của lý thuyết stream được trình bày chi tiết trong *tiểu mục 1.9*. Từ đây, luận án rút ra nền tảng lý thuyết stream dùng cho stream dạng BDL trong CS được trình bày chi tiết trong *tiểu mục 1.10*. Từ đó, luận án đi đến những nhận xét và đặt ra 3 vấn đề nghiên cứu của luận án được trình bày chi tiết trong *tiểu mục 1.11*.

1.9. Thế mạnh và hạn chế của lý thuyết stream

Lý thuyết stream cung cấp một khung toán học hình thức chặt chẽ để xử lý stream dạng BDL, hỗ trợ định nghĩa và phân tích các tính chất của stream. Tuy nhiên, do tính trừu tượng cao và độ phức tạp lớn nên việc áp dụng lý thuyết này vào thực tiễn khó khăn. Quá trình chuyển từ mô hình lý thuyết sang tính toán cụ thể còn nhiều trở ngại do thiếu công cụ hỗ trợ. Mặc dù mạnh mẽ, lý thuyết stream hiện vẫn chủ yếu ứng dụng trong hình thức hóa và cấu trúc dữ liệu vô hạn động [CT.8].

1.9.1. Thế mạnh của đại số stream

Đại số stream là một khung toán học hình thức để định nghĩa và thao tác trên các stream dạng BDL, giúp đảm bảo tính đúng đắn cho các thuật toán và hệ thống sử dụng stream. Nó cho phép xây dựng các stream phức tạp từ các stream đơn giản nhờ tính liên kết, hỗ trợ hình thức hóa đáp ứng mô hình hóa, phân tích tính chất và kiểm chứng hình thức. Đại số stream có khả năng biểu đạt linh hoạt, với nhiều phép toán số học, các phép toán xử lý stream và biến đổi trên các stream, giúp dễ dàng diễn đạt các thao tác cần thực hiện. Khung này đặc biệt phù hợp cho các stream dạng BDL, thích hợp với các ứng dụng như xử lý tín hiệu, hệ thống thời gian thực, và nghiên cứu lý thuyết ngôn ngữ hình thức [CT.8].

1.9.2. Hạn chế của đại số stream

Đại số stream có tính trừu tượng cao, gây khó khăn cho người không quen với toán học nâng cao và làm hạn chế khả năng ứng dụng. Việc chuyển đặc tả lý thuyết thành triển khai rất phức tạp, do cấu trúc lý thuyết không tương thích với mã thực tế. Công cụ hỗ trợ còn hạn chế so với các phương pháp lập trình chính thống, khiến thực nghiệm khó khăn. Đại số stream hữu ích trong các lĩnh vực như xử lý tín hiệu và hình thức hóa, nhưng ứng dụng cho dữ liệu hữu hạn còn hạn chế, nhất là trong môi trường thời gian thực vẫn là thách thức [CT.8].

1.9.3. Thế mạnh của đồng đại số stream

Đồng đại số stream là một khung toán học sử dụng phương pháp đồng quy nạp để định nghĩa và lập luận về các stream, đặc biệt hiệu quả với các cấu trúc dữ liệu vô hạn. Phương pháp này tập trung vào hành vi hệ thống theo thời gian thực, phù hợp để mô hình hóa các hệ thống động và liên tục biến đổi. Nó hỗ trợ hình thức hóa stream, cho phép mô tả hành vi phức tạp và thúc đẩy tái sử dụng qua các cấu trúc biểu đạt linh hoạt. Đặc biệt, tính đối ngẫu giữa đại số stream và đồng đại số stream mang lại những cách tiếp cận hữu ích trong xử lý dữ liệu vô hạn [CT.8].

1.9.4. Hạn chế của đồng đại số stream

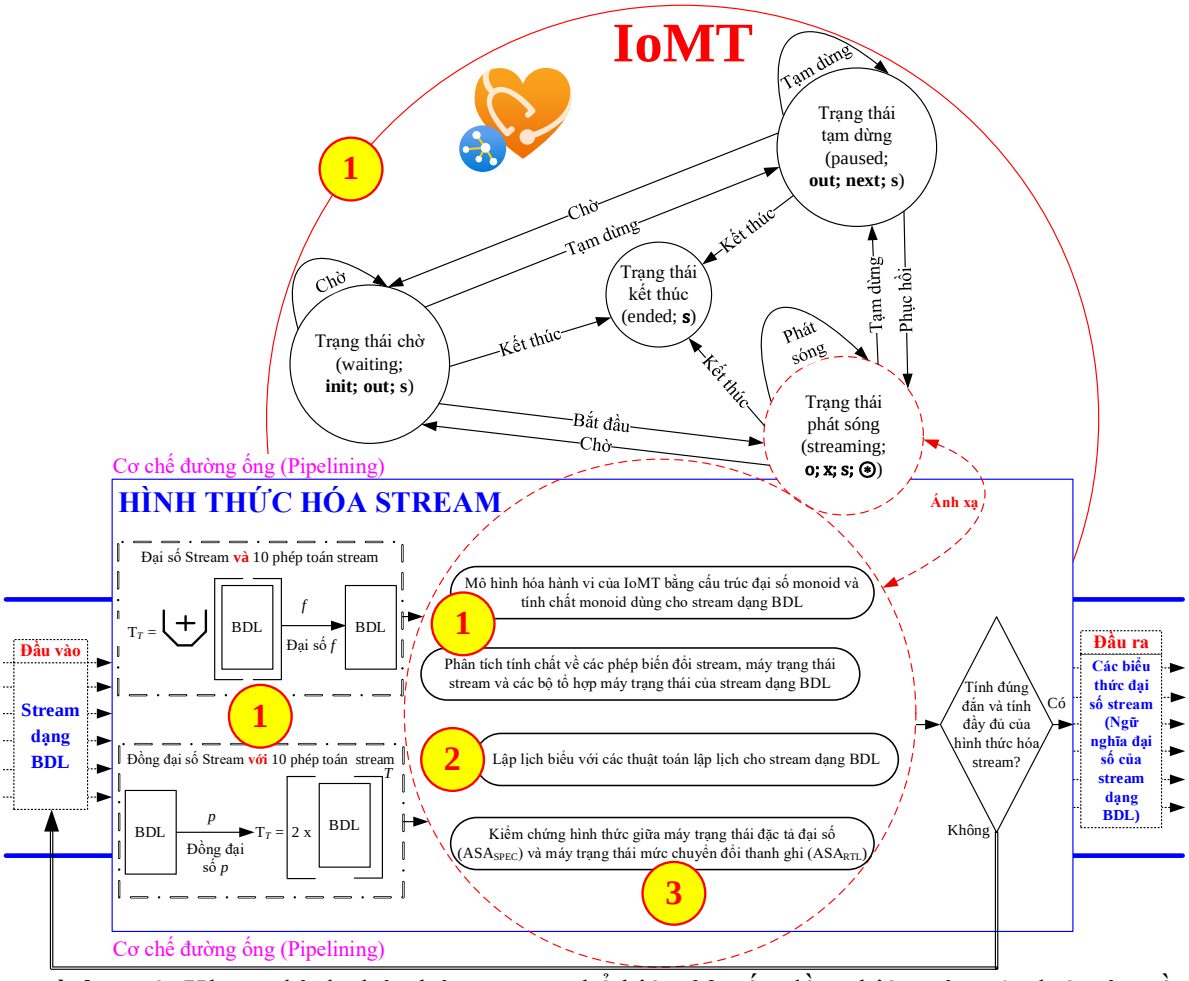
Bản chất đồng quy nạp và các khái niệm trừu tượng trong đồng đại số stream có thể khó hiểu. Việc chuyển đổi đặc tả đồng đại số stream thành triển khai thực tế gặp nhiều thách thức, vì tính trừu tượng của nó không dễ dàng áp dụng vào mô hình lập trình thực tiễn. Dù có công cụ cho đồng quy nạp, hệ sinh thái công cụ cho đồng đại số stream vẫn chưa phát triển, cản trở việc áp dụng và thử nghiệm. Tuy nhiên, nó rất phù hợp để mô hình hóa các hệ thống vô hạn, kiểm chứng hình thức và một số loại stream dữ liệu. Khả năng áp dụng đồng đại số stream cho các vấn đề dữ liệu hữu hạn còn hạn chế. Cách hình thức và ký hiệu trong đồng đại số stream có thể khó hiểu, yêu cầu hiểu biết sâu về đồng quy nạp và toán học [CT.8].

1.10. Lý thuyết stream dùng cho stream dạng BDL

Lý thuyết stream trong bối cảnh dữ liệu lớn xoay quanh việc xử lý stream dạng BDL liên tục theo thời gian thực. Việc áp dụng lý thuyết stream vào phân tích stream dạng BDL liên quan đến một số nguyên lý và hình thức hóa để xử lý stream dữ liệu dạng BDL liên tục, một tiếp cận hình thức hóa hiệu quả để đáp ứng mô hình hóa và kiểm chứng hình thức cơ chế hoạt động của stream dạng BDL [CT.8].

1.11. Nhận xét và đặt những vấn đề nghiên cứu

Việc tổng quan phân tích dữ liệu, những lý thuyết nền tảng, và tổng quan các nghiên cứu [9-21] đã đưa ra các phân tích và tính toán stream về *dãy số nguyên, số thực, mạch số thực, mạch số nguyên* thuộc $\mathbb{N}^{\omega}$ và $\mathbb{R}^{\omega}$. Ngoài ra, các mô hình stream dữ liệu truyền thống đều xem stream như là luồng ổn định, không mô hình hóa tốt tính xen kẽ frame, tốc độ luồng thay đổi, và cấu trúc đa phương thức (văn bản, hình ảnh, âm thanh, video). Dựa trên cơ sở đó, luận án tập trung vào khảo sát, phân tích và nghiên cứu tập các stream $\mathbb{A}^{\omega}$ dạng BDL bằng việc đặt ra những vấn đề nghiên cứu ở *tiểu mục 1.11.1*.



Hình 1.10. Khung hình thức hóa stream thể hiện 03 vấn đề nghiên cứu của luận án về hình thức hóa cơ chế hoạt động của stream BDL trong IoMT

1.11.1. Đặt những vấn đề nghiên cứu

Trong lĩnh vực xử lý stream, mặc dù lý thuyết stream đã có những đóng góp đáng kể nhưng về tính sẵn có của hình thức hóa stream dạng BDL vẫn còn bỏ ngỏ. Luận án đặt ra những vấn đề nghiên cứu dựa vào phương pháp hình thức để hình thức hóa cơ chế hoạt động của stream dạng BDL trong thời gian thực. Với những phân tích trên, luận án đặt ra 3 vấn đề nghiên cứu xây dựng một số khía cạnh đại số dùng cho stream dạng BDL hình 1.10.

1.11.2. Vấn đề nghiên cứu số 1

Góp phần giải quyết thách thức về tính trừu tượng của mô hình hóa stream dạng BDL, luận án nghiên cứu xây dựng một số khía cạnh đại số, gồm: xây dựng mười phép toán xử lý stream *(drop, take, zip, split, reverse, merge, convolution product, copy, register, assign)*, mở rộng đại số stream và đồng đại số stream, xây dựng cấu trúc đại số monoid, và tính chất monoid dùng cho stream dạng BDL trong IoMT. Một số khía cạnh này giúp xác định ngữ nghĩa đại số của stream dưới dạng các biểu thức đại số. Vấn đề nghiên cứu số ① này sẽ được trình bày chi tiết trong chương 2.

1.11.3. Vấn đề nghiên cứu số 2

Góp phần giải quyết thách thức về phân tích tính chất của mô hình hóa stream dạng BDL, luận án nghiên cứu xây dựng cơ chế lập lịch để xử lý stream dạng BDL gồm 04 bước: Phân hoạch các phép toán stream; Lập lịch cho tính toán BDL; Cấp phát và liên kết thanh ghi; Cấp phát và liên kết đơn vị chức năng. Việc lập lịch này nhằm mang lại khai thác các lợi ích cốt lõi như: giảm độ trễ (bằng việc chia các CStep), tăng thông lượng (bằng việc lập lịch theo cơ chế đường ống), cân bằng tải (bằng việc phân bố đều các phép toán stream trên từng CStep), và tối ưu hóa tài nguyên (bằng việc tái sử dụng thanh ghi tại mỗi CStep). Vấn đề nghiên cứu số ② này sẽ được trình bày chi tiết trong chương 3.

1.11.4. Vấn đề nghiên cứu số 3

Góp phần giải quyết về thách thức về khả năng kiểm chứng hình thức stream dạng BDL, luận án nghiên cứu xây dựng mô hình ngữ nghĩa đại số (ASM) để xử lý stream dạng BDL. Mô hình này cho phép đặc tả mối quan hệ giữa máy trạng thái mức chuyển đổi thanh ghi (ASA_{RTL}) và máy trạng thái đặc tả đại số (ASA_{SPEC}) để kiểm chứng các kết quả tổng hợp RTL và SPEC. Việc kiểm chứng này mang lại sự khẳng định $ASA_{RTL} = ASA_{SPEC}$ là đúng hay không ($ASA_{RTL} \subset ASA_{SPEC}$ hoặc $ASA_{RTL} \supset ASA_{SPEC}$). Vấn đề nghiên cứu số ③ này sẽ được trình bày ở chương 4.

Ba mục tiêu nghiên cứu của luận án được thiết kế theo hướng bổ trợ lẫn nhau, tạo thành một chỉnh thể logic và nhất quán, qua đó bao quát các vấn đề cốt lõi còn chưa được giải quyết trong bài toán đại số stream dùng cho khoa học phân tích dữ

liệu lớn dạng livestream. Sơ đồ minh họa ở Hình 1.10 thể hiện rõ cấu trúc liên kết chặt chẽ giữa các chương, phản ánh sự xuyên suốt trong định hướng nghiên cứu và triển khai nội dung của luận án.

1.12. Phần kết chương 1

Chương này đã trình bày những nghiên cứu về dữ liệu lớn có dữ liệu lớn dạng livestream (BDL) và các khái niệm cơ bản của lý thuyết stream làm cơ sở hình dung các công việc nghiên cứu tiếp theo của luận án. Phụ thuộc vào nhu cầu khám phá những đặc tính khoa học của stream dạng BDL, luận án tiến hành phân tích và so sánh các đặc trưng của lý thuyết stream, từ đó rút ra những thế mạnh và hạn chế của lý thuyết này làm nền tảng áp dụng vào luận án.

Với bốn thách thức về hình thức hóa stream dạng BDL được luận án chỉ ra ở phần *Mở đầu* trong *Tiểu mục 2*, để giải quyết bốn thách thức này, chương 1 đã giới thiệu một số lý thuyết nền tảng cần thiết để áp dụng vào xây dựng một số khía cạnh đại số, tối ưu hóa lập lịch, và mô hình ngữ nghĩa đại số để hình thức hóa cơ chế hoạt động của stream nhằm đáp ứng ngữ nghĩa đại số của stream dạng BDL. Những lý thuyết nền tảng bao gồm:

- Dữ liệu, dữ liệu frame, hàm, phép toán tích và tổng, tập hợp con, quan hệ, cấu trúc monoid, quan hệ thứ tự và thứ tự cục bộ, cấu xạ với phạm trù, hàm tử, hàm tử trên tập hợp, đồng cấu, các phép toán stream cơ bản, khái niệm về stream, đồng quy nạp, mô phỏng hai chiều, phương trình vi phân, đại số stream, đồng đại số stream, những lý thuyết nền tảng này được sử dụng để phục vụ cho *chương 2*.
- Cơ chế đường ống, dữ liệu, dữ liệu frame, hàm, ngôn ngữ RTL, cấu xạ, cơ chế lập lịch, bước điều khiển, các phép toán stream cơ bản, khái niệm về stream, đồng quy nạp, mô phỏng hai chiều, phương trình vi phân, đại số stream, đồng đại số stream, những lý thuyết nền tảng này được sử dụng để phục vụ cho *chương 3*.
- Dữ liệu, dữ liệu frame, hàm, ngôn ngữ RTL, không gian Chu, bước điều khiển, một số quan hệ nền tảng giữa trạng thái và sự kiện trong không gian Chu, những lý thuyết nền tảng này được sử dụng để phục vụ cho *chương 4*.

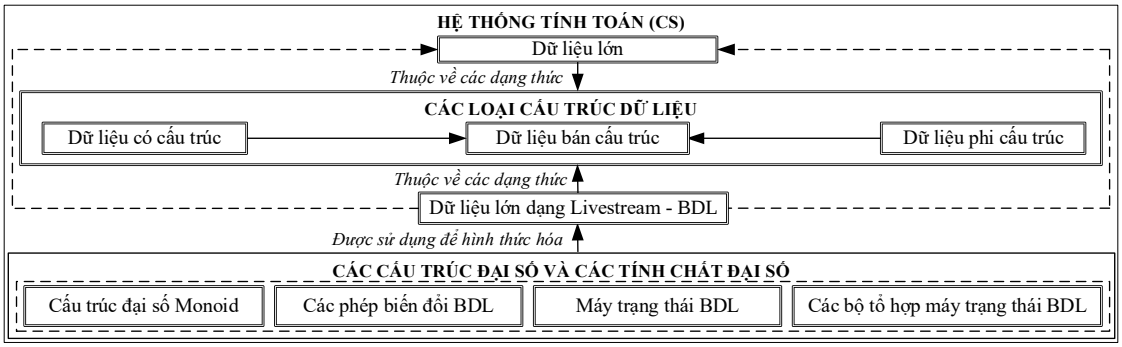
Chương 2 xây dựng một số khía cạnh đại số, gồm: Xây dựng mười phép toán xử lý stream, tổ hợp các phép toán này thành các biểu thức stream, mở rộng đại số stream và đồng đại số stream, áp dụng mười phép toán xử lý stream này vào xây dựng cấu trúc đại số monoid và các tính chất monoid để hình thức hóa cơ chế hoạt động của stream dạng BDL trong IoMT.

Chương 2. XÂY DỰNG MỘT SỐ KHÍA CẠNH ĐẠI SỐ CỦA STREAM DẠNG BDL TRONG IoMT

Chương này tập trung xây dựng một số khía cạnh đại số: Xây dựng mười phép toán xử lý stream, mở rộng đại số stream và đồng đại số stream, xây dựng cấu trúc đại số monoid và tính chất monoid dùng cho stream dạng BDL trong IoMT. Đóng góp này để hình thức hóa (mô hình hóa, phân tích tính chất và kiểm chứng hình thức) cơ chế hoạt động của stream nhằm đáp ứng ngữ nghĩa đại số của stream dạng BDL. Cấu trúc đại số monoid hình thức hóa cơ chế hoạt động của stream bằng mười phép toán xử lý stream được luận án xây dựng áp dụng vào tính chất kết hợp và tính chất đồng nhất stream dạng BDL. Tính chất monoid để hình thức hóa cơ chế hoạt động của stream bằng các phép biến đổi stream, máy trạng thái stream, và các bộ tổ hợp máy trạng thái stream dạng BDL trong IoMT. Những kết quả nghiên cứu của chương 2 này đã được công bố trong công trình [CT.2] và [CT.3].

2.1. Giới thiệu hệ thống tính toán

Theo dòng thời gian, sự xuất hiện đa dạng của các thiết bị điện tử (máy tính, điện thoại, sensor, IoT và IoMT) đang từng giây tạo ra khối lượng dữ liệu lớn (bao gồm stream dạng BDL) cùng với các tính chất phức tạp của nó trong các hệ thống tính toán (CS) [CT.1][CT.3][CT.5][CT.7][75, 76] (Hình 1.1, 1.6 và 2.1). Nhận thấy, trên các nền tảng mạng xã hội hiện nay đang có lưu lượng livestream rất lớn, đây là cơ sở để luận án tập trung vào nghiên cứu một số khía cạnh đại số của stream để hình thức hóa cơ chế hoạt động của stream dạng BDL trong CS nhằm đáp ứng ngữ nghĩa đại số của stream (Hình 2.1).



Hình 2.1. Hình thức hóa stream dạng BDL trong các hệ thống tính toán (CS) [CT.7]

Do thách thức về tính sẵn có của hình thức hóa stream và xuất phát từ tính chất tự nhiên của stream dạng BDL (Hình 2.1), chương này xây dựng mười phép toán xử lý stream [CT.1][CT.2], cấu trúc đại số monoid và các tính chất monoid để hình thức hóa cơ chế hoạt động của stream. Các tính chất monoid tạo ra một số khía cạnh đại số gồm: các phép biến đổi stream, máy trạng thái stream dạng BDL và các bộ tổ hợp của máy trạng thái stream dạng BDL trong IoMT [CT.3][CT.5][CT.7].

2.2. Phân tích IoMT, BDL và mối quan hệ giữa chúng

Những tiến bộ về công nghệ (Sensor, IoT, IoMT, CS) cho phép chúng ta gửi stream dạng BDL đã thu thập được từ các công nghệ này đến các trung tâm tính toán để xử lý trong thời gian thực. IoMT là một mạng kết nối các thiết bị với nhau [77, 78] gồm: bộ truyền động, bộ cảm biến, máy ảnh và nhiều thiết bị khác tạo ra lượng lớn dữ liệu. Do sự phát triển quy mô lớn của các ứng dụng trong IoMT làm cho việc sản xuất dữ liệu lớn lên đến geophytes (2^{10} brontobytes) và vượt qua cả brontobytes (2^{10} yottabytes) (Hình 1.6). Các ứng dụng này gồm: thành phố thông minh, công nghiệp thông minh, nhà thông minh, hệ thống vận tải thông minh [28], và hệ thống cảm biến thông minh đã tạo ra lượng lớn dữ liệu như vậy. Do vậy, mối quan hệ giữa IoMT và BDL đóng một vai trò quan trọng trong sự tiến bộ của cuộc cách mạng kỹ thuật và công nghệ [76, 79].

2.2.1. Khái niệm IoMT

Hiện nay, ý tưởng về kết nối tất cả vạn vật di động dẫn đến sự xuất hiện của IoT và IoMT, có thể được sử dụng như một thiết bị di động để thu thập dữ liệu từ các thiết bị IoMT (Hình 1.6 trong chương 1) được triển khai từ xa hoặc có thể hoạt động như một cầu nối giao tiếp [28, 77]. Như vậy, IoMT đang thay đổi cơ bản thế giới bằng cách cho phép nhiều thiết bị di động giao tiếp và trao đổi dữ liệu với nhau để rồi dữ liệu được sinh ra một cách tự nhiên. Người dùng có thể được xem như một loại đối tượng di động, họ được trang bị các thiết bị đeo có thể cung cấp hoặc nhận các dịch vụ từ IoMT hoặc đến IoMT [80]. Bộ các vật thể di động có thể tạo ra mạng không dây di động và cung cấp các dịch vụ. Mạng này có thể bao phủ một khu vực rộng lớn, trong khi các thiết bị kết nối với mạng tương tác lẫn nhau để thực hiện các nhiệm vụ phức tạp [81].

Dựa vào phân tích ở trên và áp dụng phương pháp hình thức (Luận án giả định xem IoMT là mềm hóa), IoMT được luận án hình thức hóa chung cho một tập các trạng thái hữu hạn $x \in IoMT$, mỗi trạng thái có thể nhận biết được là đang ở trong IoMT (Khi đó, ký hiệu $IoMT$ được sử dụng trong công thức toán học hoặc biểu thức toán học), sao cho mỗi cặp trạng thái $x, y \in IoMT$ có thể xác định nếu $x = y$ hoặc không. Ký hiệu $\emptyset$ biểu thị *IoMT* không có trạng thái nào. Nếu $IoMT_1$ và $IoMT_2$ là những IoMT, đặc tả $IoMT_1$ là một hệ thống con của $IoMT_2$ được viết $IoMT_1 \subseteq IoMT_2$, nếu mọi trạng thái của $IoMT_1$ là một trạng thái của $IoMT_2$. Như vậy, với bất kỳ hệ thống *IoMT* nào thì đều có các hệ thống con $\emptyset \subseteq IoMT$ và $IoMT \subseteq IoMT$.

Luận án sử dụng các ký hiệu của hệ thống tính toán ở trên để biểu diễn hệ thống con. Cho *IoMT* được viết $\{x \in IoMT \mid x \text{ là một trạng thái của } IoMT\}$. Ký hiệu $\exists$ có

nghĩa là biểu đạt *"tồn tại"*. Do đó, có thể viết IoMT dưới dạng biểu thức toán học $\{x \in IoMT \mid \exists y$ là một trạng thái cuối sao cho $BDL(x) = y\}$. Ký hiệu $\exists!$ có nghĩa là *"tồn tại duy nhất"*. Vì vậy, phát biểu "$\exists! x \in IoMT$ là một trạng thái ban đầu" có nghĩa là có một và chỉ một trạng thái để bắt đầu, được hiểu là trạng thái của *IoMT* trước khi thực hiện bất kỳ hoạt động BDL nào được xử lý. Ký hiệu $\forall$ có nghĩa là *"với mọi"*. Vì vậy, phát biểu "$\forall x \in IoMT, \exists y \in IoMT$ sao cho $BDL(x) = y$" có nghĩa là với mọi trạng thái x của IoMT, tồn tại một trạng thái y tiếp theo. Ký hiệu $\stackrel{\text{def}}{=}$ được dùng để đặc tả IoMT như "$IoMT_1 \stackrel{\text{def}}{=} IoMT_2$" nghĩa là *"đặc tả $IoMT_1$ là $IoMT_2$"*.

2.2.2. Thành phần của BDL trong IoMT

Nếu $IoMT_1$ và $IoMT_2$ là tập hợp các trạng thái của IoMT, khi đó một hành động BDL từ $IoMT_1$ đến $IoMT_2$ được viết $BDL: IoMT_1 \to IoMT_2$ là một ánh xạ gửi từng trạng thái $x \in$ IoMT$_1$ đến từng trạng thái của $IoMT_2$, được ký hiệu $BDL(x) \in IoMT_2$. Luận án gọi $IoMT_1$ là miền của BDL và gọi $IoMT_2$ là đồng miền của BDL.

Với mỗi trạng thái $x \in IoMT_1, IoMT_2$, có đúng một ánh xạ bắt nguồn từ x, nhưng đối với một trạng thái $y \in IoMT_2$, có thể có nhiều ánh xạ đến trạng thái y hoặc có thể không có ánh xạ nào đến y.

Giả sử $IoMT_2 \subseteq IoMT_1$ là một hệ thống tính toán con. Khi đó, có thể xem xét hành động BDL của $IoMT_2 \to IoMT_1$ bằng cách gửi từng trạng thái của $IoMT_2$ đến chính nó như là một trạng thái của $IoMT_1$. Nếu $IoMT_1 = \{a, b, c, d, e, f\}$ và $IoMT_2 = \{b, d, e\}$, thì $IoMT_2 \subseteq IoMT_1$ và chuyển đổi nó thành hành động BDL của $IoMT_2 \to IoMT_1$, với $b \mapsto b, d \mapsto d, e \mapsto e$. Ký hiệu $\mapsto$ này đọc thành *"được ánh xạ đến"*. Một hành động BDL có dạng $BDL: IoMT_1 \to IoMT_2$ nghĩa là một quy tắc gán từng trạng thái $x \in IoMT_1$ cho từng trạng thái $BDL(x) \in IoMT_2$. Do vậy, đặc tả "x được ánh xạ đến $BDL(x)$" và được viết dưới dạng $x \mapsto BDL(x)$.

Khái niệm về hệ thống con, luận án đặc tả $BDL: IoMT_2 \subseteq IoMT_1$ như là một hệ thống tính toán con. Ở đây, cần làm rõ $IoMT_2$ là một hệ thống tính toán con của $IoMT_1$, nhưng BDL là tên của hành động BDL liên quan. Cho một hành động BDL có dạng $BDL: IoMT_1 \to IoMT_2$, các trạng thái của $IoMT_2$ có ít nhất một ánh xạ đến chúng được gọi là ảnh của BDL *(Tiểu mục 1.6.1, Mục c), Chương 1)*, suy ra ta có ảnh của BDL như sau:

- $\text{image}(BDL) \stackrel{\text{def}}{=} \{y \in IoMT_2 \mid \exists x \in IoMT_1$ sao cho $BDL(x) = y\}$. (2.1)

Cho $BDL_1: IoMT_1 \to IoMT_2$ và $BDL_2: IoMT_2 \to IoMT_3$, trong đó cùng miền giá trị của BDL_1 là cùng một tập hợp các trạng thái của $IoMT_1$ như miền đầu vào của BDL_2 (nói cách khác là $IoMT_2$), ta đặc tả BDL_1 và BDL_2 có thể là sự liên kết hoặc ghép nối theo cách sau: $IoMT_1 \xrightarrow{BDL_1} IoMT_2 \xrightarrow{BDL_2} IoMT_3$.

Việc liên kết giữa BDL_1 và BDL_2 được ký hiệu $BDL_2 \circ BDL_1: IoMT_1 \to IoMT_3$.

Sử dụng ký hiệu $\mathrm{Hom}_{\mathrm{IoMT}}(IoMT_1, IoMT_2)$ để chỉ tập hợp hành động các BDL có dạng $IoMT_1 \to IoMT_2$. Hai hành động BDL là BDL_1 và $BDL_2: IoMT_1 \to IoMT_2$ bằng nhau nếu và chỉ nếu với mọi trạng thái $x \in IoMT_1$ thì ta có $BDL_1(x) = BDL_2(x)$. Định nghĩa BDL đơn vị trên $IoMT$ được ký hiệu là $1_{IoMT}: IoMT \to IoMT$, được gọi là hành động BDL sao cho với mọi trạng thái x có dạng $x \in IoMT$ dẫn đến $1_{IoMT}(x) = x$ *(Tiểu mục 1.6.1, Mục c), Chương 1)*.

Định nghĩa 2.1 (Đồng cấu của IoMT): Một hành động $BDL_1: IoMT_1 \to IoMT_1$ được gọi là một đồng cấu, ký hiệu là: $BDL_1: IoMT_1 \overset{\cong}{\to} IoMT_1$, nếu tồn tại một hành động $BDL_2: IoMT_2 \to IoMT_1$ sao cho $BDL_2 \circ BDL_1 = 1_{IoMT_1}$ và $BDL_1 \circ BDL_2 = 1_{IoMT_2}$. Khi đó, BDL_1 là đồng cấu khả nghịch, BDL_2 là nghịch đảo của BDL_1, hai đặc tả $IoMT_1$ và $IoMT_2$ được gọi là đồng cấu với nhau, ký hiệu: $IoMT_1 \cong IoMT_2$.

Trong định nghĩa 2.1, ký hiệu 1_{IoMT_2} là ký hiệu hàm đồng nhất trên tập $X = IoMT_2$, cụ thể: $1_{IoMT_2}: IoMT_2 \to IoMT_2$ là ánh xạ sao cho với mọi trạng thái $x \in IoMT_2$, ta có: $1_{IoMT_2}(x) = x$. Tương tự, 1_{IoMT_1} là ánh xạ đồng nhất trên $IoMT_1$. Vai trò trong định nghĩa, đồng cấu khả nghịch $BDL_1: IoMT_1 \to IoMT_2$ và $BDL_2: IoMT_2 \to IoMT_1$, với điều kiện $BDL_2 \circ BDL_1 = 1_{IoMT_1}$ và $BDL_1 \circ BDL_2 = 1_{IoMT_2}$, tức là hai ánh xạ này là nghịch đảo của nhau, đảm bảo rằng có thể đi tới và quay về mà không mất thông tin, một điều kiện chuẩn của đồng cấu trong lý thuyết phạm trù *(Tiểu mục 1.6.1, Mục j) và Mục n), và Định nghĩa 1.3, Chương 1)*.

Mệnh đề 2.1: Các yếu tố sau đây đúng với sự đồng cấu của IoMT.

- Với bất kỳ một IoMT nào, *IoMT* đều là đồng cấu với chính nó, tức là có tồn tại một IoMT đồng cấu với nó được viết $IoMT \overset{\cong}{\to} IoMT$.
- Với bất kỳ hai IoMT nào $IoMT_1$ và $IoMT_2$, nếu $IoMT_1$ là đồng cấu với $IoMT_2$ thì $IoMT_2$ cũng đồng cấu với $IoMT_1$.
- Với bất kỳ ba IoMT nào $IoMT_1$, $IoMT_2$ và $IoMT_3$, nếu $IoMT_1$ là đồng cấu với $IoMT_2$ và $IoMT_2$ là đồng cấu với $IoMT_3$ thì $IoMT_1$ cũng là đồng cấu với $IoMT_3$.

Chứng minh: Mệnh đề 2.1 có các yếu tố đúng đắn về đồng cấu sau đây

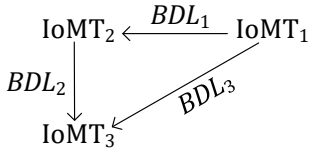
- Hành động BDL đồng nhất $1_{IoMT}: IoMT \to IoMT$ có thể đảo ngược, nghịch đảo của nó là 1_{IoMT} vì $1_{IoMT} \circ 1_{IoMT} = 1_{IoMT}$.
- Nếu $BDL_1: IoMT_1 \to IoMT_2$ có thể đảo ngược với nghịch đảo là $BDL_2: IoMT_2 \to IoMT_1$ thì BDL_2 là một đồng cấu với nghịch đảo là BDL_1.
- Nếu $BDL_1: IoMT_1 \to IoMT_2$ và $\widehat{BDL_1}: IoMT_2 \to IoMT_3$ đều có thể đảo ngược với các nghịch đảo là $BDL_2: IoMT_2 \to IoMT_1$ và $\widehat{BDL_2}: IoMT_3 \to IoMT_2$ thì các tính toán sau đây cho thấy $\widehat{BDL_1} \circ BDL_1$ cũng có thể đảo ngược với nghịch đảo là $BDL_2 \circ \widehat{BDL_2}$: $(\widehat{BDL_1} \circ BDL_1) \circ (BDL_2 \circ \widehat{BDL_2}) = \widehat{BDL_1} \circ (BDL_1 \circ BDL_2) \circ \widehat{BDL_2} = BDL_1 \circ 1_{IoMT_2} \circ \widehat{BDL_2} = \widehat{BDL_1} \circ \widehat{BDL_2} = 1_{IoMT_3}$ và $(BDL_2 \circ \widehat{BDL_2}) \circ (\widehat{BDL_1} \circ BDL_1) = BDL_2 \circ$

$(\widehat{BDL_2} \circ \widehat{BDL_1}) \circ BDL_1 = BDL_2 \circ 1_{IoMT_2} \circ BDL_1 = BDL_2 \circ BDL_1 = 1_{IoMT_1}$. □

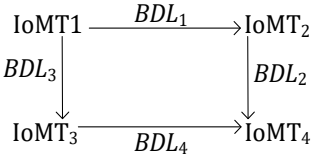
2.2.3. Tính chất giao hoán của BDL trong IoMT

Hình 2.2 là một sơ đồ của các IoMT, nếu mỗi $IoMT_1$, $IoMT_2$ và $IoMT_3$ là một IoMT và mỗi BDL_1, BDL_2 và BDL_3 là một hành động BDL. Như vậy, sơ đồ ở hình 2.2 là một giao hoán nếu $BDL_2 \circ BDL_1 = BDL_3$ và gọi nó là một tam giác giao hoán của các IoMT. Như vậy, xét sơ đồ ở hình 2.2 và sơ đồ ở hình 2.4 sau đây:

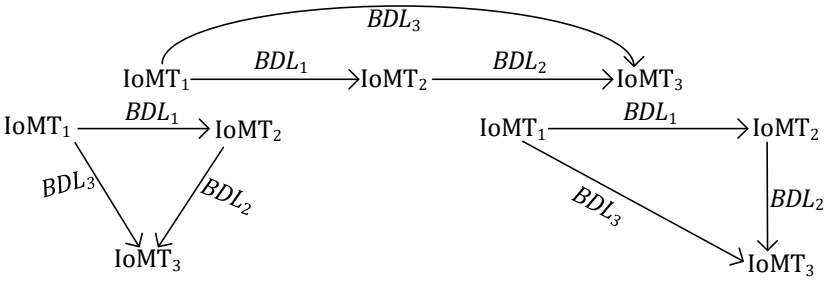
- Sơ đồ ở hình 2.4 được mô tả theo hai khía cạnh sau:
  - Giữa sơ đồ ở hình 2.2 và hình 2.4 giống nhau về mặt ngữ nghĩa hoạt động;
  - Giữa sơ đồ ở hình 2.2 và hình 2.4 khác nhau về cách thức biểu diễn.



Hình 2.2. Sơ đồ hình tam giác giao hoán của các IoMT



Hình 2.3. Sơ đồ hình chữ nhật giao hoán của các IoMT



Hình 2.4. Sơ đồ giao hoán của các IoMT

Dựa trên cơ sở các phân tích hình thức ở trên, sơ đồ ở hình 2.3 của các IoMT như sau: Nếu mỗi $IoMT_1$, $IoMT_2$, $IoMT_3$ và $IoMT_4$ là một IoMT và mỗi BDL_1, BDL_2, BDL_3 và BDL_4 là một hành động BDL. Do vậy, đặc tả sơ đồ ở hình 2.3 có tính chất giao hoán nếu $BDL_2 \circ BDL_1 = BDL_4 \circ BDL_3$ và gọi nó là một hình chữ nhật giao hoán của các IoMT.

2.3. Đại số stream và đồng đại số stream dùng cho stream dạng BDL

2.3.1. Đại số stream dùng cho phân tích stream dạng BDL

Dựa trên nền tảng đại số stream *(Tiểu mục 1.6.8, Chương 1)*, luận án xây dựng mười phép toán xử lý stream dạng BDL trong công trình [CT.2]: dropping ($drop$), taking ($take$), zipping (zip), splitting ($split$), reverse (rev), merging ($merge$), (convolution) product ($\times$), copying ($\bullet$), registering ($\blacksquare$), và assignment ($:=$).

Luận án phân tích thành hai loại phép toán xử lý stream dùng cho BDL như sau: Phép toán xử lý stream một ngôi có dạng $\mathbb{A}^\omega \to \mathbb{A}^\omega$, được ký hiệu là $\boxdot \in \{drop, take, zip, split, rev\}$ bằng phương trình sau: $\boxdot: \sigma \in \mathbb{A}^\omega \to \boxdot\,(\sigma) \in \mathbb{A}^\omega$. Phép toán xử lý stream hai ngôi có dạng $\mathbb{A}^\omega \times \mathbb{A}^\omega \to \mathbb{A}^\omega$, được ký hiệu là $\circledast \in \{+, \times, :=\}$ bằng phương trình sau: $\circledast: \sigma \in \mathbb{A}^\omega \times \tau \in \mathbb{A}^\omega \to \sigma \circledast \tau \in \mathbb{A}^\omega$ [CT.2]. Từ

đây, luận án đề xuất xây dựng một hàm stream dạng BDL [CT.2] như sau (2.2):

$$\sigma: \mathbb{N} \to \mathbb{A}^{\omega} \quad (2.2)$$

Trong đó:

- σ là một stream dạng BDL.
- $\sigma(0)$, $\sigma(1)$, $\sigma(2)$, ... là các frame.
- $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$ là tập hợp các số tự nhiên.
- $\mathbb{A}^{\omega}$ là tập các stream dạng BDL.
- $\mathbb{A}$ là tập các frame. Một frame được định nghĩa như một cấu trúc toán học chứa các loại dữ liệu như: văn bản (text), hình ảnh (image), âm thanh (audio), video, ánh sáng, không gian, thời gian, v.v.

Các khái niệm về stream dùng cho BDL đã được luận án khảo sát và phân tích trong Chương 1. Luận án trình bày một số lập luận về cấu trúc stream dùng cho BDL được phân tích (2.2) như sau:

- *Giá trị khởi đầu của stream $\sigma(0)$ và đạo hàm của stream σ':* Điều này tương ứng với đầu (head) và đuôi (tail) của một stream. Giá trị khởi đầu là phần tử đầu tiên của stream và đạo hàm của stream là phần còn lại của stream sau giá trị khởi đầu.
- *Hành vi của stream:* Hành vi gồm có hai khía cạnh. *Khía cạnh một*, cho phép kiểm tra giá trị khởi đầu; *Khía cạnh hai*, có thể phát triển một stream mới có dạng σ', bao gồm stream khởi đầu nhưng lại loại bỏ phần tử đầu tiên, tức là $\sigma'(0) = \sigma(1)$.
- *Phương trình vi phân hành vi stream:* Xác định một stream bằng cách biểu diễn giá trị khởi đầu $\sigma(0)$, biểu diễn đạo hàm stream với $n \geq 0$, phương trình vi phân hành vi stream có dạng sau: $\left(\sigma^{(n)}\right)' = \sigma^{(n+1)}$. Biểu diễn đạo hàm stream là cách cho ta biết cách để tiếp tục tạo stream mới sau giá trị stream khởi đầu.

Định lý 2.1: $\forall \sigma \in \mathbb{A}^{\omega}$ khi đó ta có phương trình stream $\sigma = \sigma(0) + (X \times \sigma')$.

Chứng minh: Một stream σ được biểu đạt là stream dạng BDL $[\sigma(0)] = (\sigma(0), 0, 0, \ldots)$ và thêm vào đó là $X \times \sigma' = X \times (\sigma(1), \sigma(2), \sigma(3), \sigma(4), \ldots) = (0, \sigma(1), \sigma(2), \sigma(3), \ldots)$ có thể biểu diễn như sau $(0, \sigma(1), \sigma(2), \sigma(3), \ldots)$. Về mặt tương đương có thể được chứng minh bằng tiếp cận đồng quy nạp. □

Định lý 2.1 ở trên là tương đương với định lý 269 trong [20]. Trong đó, $\mathbb{A}^{\omega}$ là tập các stream dạng BDL *(Tiểu mục 1.6.1, Mục b), Chương 1)* được dùng để thay thế cho $\mathbb{R}^{\omega}$ là tập các stream dạng số thực *(Tiểu mục 1.6.3, Chương 1)*; Luận án áp dụng phép toán trộn merge $(+)$ thay thế cho phép toán liên kết $(+)$; X ám chỉ áp dụng phép toán độ trễ đơn vị $(\blacksquare)$(nghĩa là gắn register) với đạo hàm stream σ'.

2.3.2. Tập các stream dùng cho phân tích BDL

$\mathbb{A}^{\omega}$ là tập các stream dạng BDL (2.2) được định nghĩa dựa trên tập hợp các frame $\mathbb{A}$ dưới dạng sau [CT.1][CT.2]:

$$\mathbb{A}^{\omega} = \{\sigma \mid \sigma: \mathbb{N} \to \mathbb{A}\} \quad (2.3)$$

Trong đó:

- $\sigma, \tau, \gamma, \pi, \epsilon, \ldots$ được sử dụng để biểu thị cho các stream dạng BDL.
- Các phần tử stream được biểu thị dưới dạng $\sigma \in \mathbb{A}^{\omega}$ theo $\sigma = (\sigma(0), \sigma(1), \ldots)$.
- Đạo hàm của một stream có dạng $\sigma' = (\sigma(1), \sigma(2), \ldots)$ và giá trị khởi đầu của σ là $\sigma(0)$. Với mọi $n \geq 0$ và $\sigma \in \mathbb{A}^{\omega}$, đạo hàm bậc cao được định nghĩa dưới dạng $\sigma^{(0)} = \sigma$ và $\sigma^{(n+1)} = \left(\sigma^{(n)}\right)'$. Do đó, công thức tổng quát có dạng $\sigma^{(n)}(0) = \sigma(n)$ được sử dụng cho phân tích stream dạng BDL (Bảng 2.1).

Bảng 2.1. Phương trình vi phân stream σ dùng cho BDL

Phương trình vi phân stream (Tail)	Giá trị ban đầu (Head)	Điều kiện
$\left(\sigma^{(n)}\right)' = \sigma^{(n+1)}$	$\sigma(0)$ hoặc $\sigma^{(n)}(0) = \sigma(n)$	$n \geq 0$

Luận án phát triển phép tính stream gồm mười phép toán được xem như một vai trò nền tảng dành cho việc phân tích và hình thức hóa cơ chế hoạt động của stream dạng BDL trong CS được trình bày chi tiết trong tiểu mục 2.3.3 [CT.2].

2.3.3. Các phép toán xử lý stream dùng cho phân tích stream dạng BDL

Các hoạt động loại bỏ, nén, tách, đảo ngược, trộn, tích chập, và gán áp dụng cho biểu diễn stream dữ liệu kiểu dãy tuần tự $\mathbb{N}$ và $\mathbb{R}$ đã được đề xuất trong những nghiên cứu được luận án khảo sát ở bảng 1.17 (Chương 1). Từ đây, luận án xây dựng mười phép toán xử lý stream dạng BDL để hình thức hóa cơ chế hoạt động của stream nhằm đáp ứng ngữ nghĩa đại số của stream dạng BDL trong CS [CT.2], mười phép toán stream này được trình bày chi tiết từ tiểu mục 2.3.3.1 đến 2.3.3.10, và những lợi ích cụ thể của các phép toán xử lý stream được trình bày chi tiết tại tiểu mục 2.3.3.11.

2.3.3.1. Phép toán Dropping

Một cách để hình thức hóa một stream con từ một stream vô hạn (2.2) như sau: σ: $\mathbb{N} \to \mathbb{A}$ được định nghĩa bằng một hàm đơn điệu (Monotonous function) f: $\mathbb{N} \to \mathbb{N}$, nếu $n < m$ khi đó thì $f(n) < f(m)$, $\forall n, m \in \mathbb{N}$. Một hàm chỉ mục $S_f(\sigma)$ như vậy được sử dụng để xác định một stream con σ vô hạn qua biểu thức $S_f(\sigma)(n) = \sigma\big(f(n)\big)$, ở đó hàm chỉ mục này nhận vào một stream con σ và một chỉ mục n, và trả về một stream con σ tại vị trí chỉ mục n, và ngược lại đối với một stream con của một stream vô hạn, ở bất kỳ stream con nào dạng σ đều xác định duy nhất một hàm đơn điệu như vậy.

Với việc gán cho bất kỳ stream nào, ở đó stream con σ được xác định bằng một hàm đơn điệu f cho trước được định nghĩa một bộ lấy mẫu stream chu kỳ có dạng S_f: $\mathbb{A}^{\omega} \to \mathbb{A}^{\omega}$, $\sigma \mapsto S_f(\sigma)$. Với các bộ lấy mẫu stream chu kỳ sao cho chúng tạo ra một stream con của stream đầu vào cho trước bằng cách chọn lặp lại các phần tử ở các khoảng cố định từ stream đầu vào và bỏ qua tất cả phần tử khác.

Qua việc mô tả một stream con bằng một hàm đơn điệu kết hợp với bộ lấy mẫu

stream chu kỳ là các hàm chỉ mục đơn điệu ở trên. Như vậy, để chứng minh sự liên kết của hai bộ lấy mẫu stream chu kỳ lặp lại một bộ lấy mẫu stream chu kỳ, luận án tiến hành định nghĩa 2.2 như sau:

Định nghĩa 2.2: Cho $k, l \in \mathbb{N}$, với $l > 1$ và $1 \leq k \leq l$, bất kỳ dãy stream k dạng $0 \leq \mathrm{n}_0 < \mathrm{n}_1 < \cdots < \mathrm{n}_{\mathrm{k}-1} < l$ đều xác định một bộ lấy mẫu stream chu kỳ có dạng $S: \mathbb{A}^\omega \to \mathbb{A}^\omega$ của chu kỳ đầu vào và kích thước khối k đầu ra được định nghĩa bằng phương trình vi phân stream: $\mathrm{S}(\sigma)^{(k)} = \mathrm{S}\left(\sigma^{(l)}\right)$ với các giá trị khởi tạo ban đầu là $\mathrm{S}(\sigma)(j) = \sigma\left(n_j\right)$ $(0 \leq j < k)$.

Như vậy, chu kỳ và kích thước khối của stream ở mức tối thiểu thì không yêu cầu. Nếu một bộ lấy mẫu stream đã có chu kỳ l và kích thước khối k thì khi đó nó cũng có chu kỳ $2l$ với kích thước khối $2k$, v.v. Phép toán dropping của hàm even và D_4^2 ở trên được cho bởi ở bảng 2.2 và 2.3 như sau:

Bảng 2.2. Phương trình vi phân của phép toán dropping cho hàm even và D_4^2

Hàm even	Phép toán dropping D_4^2
$even(\sigma)' = even(\sigma'')$, $even(\sigma)(0) = \sigma(0)$	$D_4^2(\sigma)^{(3)} = D_4^2(\sigma)^{(4)}, D_4^2(\sigma)(0) = \sigma(0)$, $D_4^2(\sigma)(1) = \sigma(1), D_4^2(\sigma)(2) = \sigma(3)$

Bảng 2.3. Phương trình vi phân stream của phép toán dropping

Phương trình vi phân stream	Giá trị ban đầu	Điều kiện
$D_l^k(\sigma)^{(l-1)} = D_l^k(\sigma)^{(l)}$	$D_l^k(\sigma)(0) = \sigma(0)$	$l > 1$ và $1 \leq k \leq l, 0 \leq j < k$

Định nghĩa 2.3: Với $l \geq 2$ và $0 \leq k < l$, phép toán dropping $D_l^k: \mathbb{A}^\omega \to \mathbb{A}^\omega$ mang nhiệm vụ loại bỏ phần tử thứ k của từng khối stream đầu vào với kích thước khối l được định nghĩa bằng phương trình vi phân stream (Bảng 2.4).

Bảng 2.4. Phương trình vi phân stream của phép toán dropping

Phương trình vi phân stream	Giá trị ban đầu	Điều kiện
$D_l^{k+1}(\sigma)' = D_l^k(\sigma')$	$D_l^{k+1}(\sigma)(0) = \sigma(0)$	$\forall l \geq 2, 0 \leq k < l-1$
$D_l^k(\sigma)' = D_l^{l-2}(\sigma'')$	$D_l^k(\sigma)(0) = \sigma(1)$	$\forall l \geq 2$

Với phép toán dropping D_4^2, định nghĩa 2.3 tương đương với định nghĩa 2.2. Một trong những lợi ích của các định nghĩa đồng quy nạp là chúng hỗ trợ các chứng minh đồng quy nạp. Tiếp theo, luận án đưa ra một minh họa về kỹ thuật chứng minh, trong đó chứng minh quy tắc hoán đổi phép toán dropping sau:

Với mọi $l \geq 1$ và $0 \leq k \leq h \leq l$, thì $D_{l+1}^h \circ D_{l+2}^k = D_{l+1}^k \circ D_{l+2}^{h+1}$, nghĩa là một quy tắc đổi chỗ việc loại bỏ phần tử trong dãy bằng phép toán dropping cho các khối khác nhau. Quy tắc này có nghĩa là cho một stream vô hạn đầu vào σ, nếu ta loại bỏ phần tử thứ h của các khối kích thước $l+1$, rồi loại bỏ phần tử thứ k của các khối kích thước $l+2$, sau đó loại bỏ phần tử thứ $h+1$ của các khối kích thước $l+2$, kết quả sẽ tương đương với việc thực hiện các phép toán dropping phần tử theo trình tự tương ứng của các khối kích thước $l+1$ và $l+2$ của phần tử thứ $h+1$ theo quy tắc

tương tự. Đây là một quy tắc quan trọng được sử dụng trong chứng minh đồng quy nạp của các tính chất liên quan đến phép toán dropping.

Để chứng minh tính đẳng cấu của quan hệ $R \subseteq \mathbb{A}^\omega \times \mathbb{A}^\omega$ được định nghĩa bằng phương trình sau đây $R = \{\langle D_{l+1}^h \circ D_{l+2}^k(\sigma), D_{l+1}^k \circ D_{l+2}^{h+1}(\sigma)\rangle \mid \sigma \in \mathbb{A}^\omega\}$. Tính đẳng cấu này được chứng minh bằng quy nạp từ $R \cup R^{-1}$ là sự mô phỏng hai chiều của stream. Một minh họa khác về phép toán dropping $D_2^0 = D_4^0 \circ D_5^2 \circ D_6^4$, đây là một trường hợp cơ bản của một quy tắc mở rộng của phép toán này. Để chứng minh điều này, tính đẳng cấu của quan hệ $R \subseteq \mathbb{A}^\omega \times \mathbb{A}^\omega$ được định nghĩa bằng phương trình: $R = \{\langle D_2^0(\sigma), D_4^0 \circ D_5^2 \circ D_6^4(\sigma)\rangle \mid \sigma \in \mathbb{A}^\omega\} \cup \{\langle D_2^0(\sigma), D_4^2 \circ D_5^0 \circ D_6^2(\sigma)\rangle \mid \sigma \in \mathbb{A}^\omega\}$ $\cup \{\langle D_2^0(\sigma), D_4^1 \circ D_5^3 \circ D_6^0(\sigma)\rangle \mid \sigma \in \mathbb{A}^\omega\}$. Tính đẳng cấu này được suy ra bằng chứng minh đồng quy nạp cho thấy rằng R là một mô phỏng hai chiều của stream.

Ví dụ 2.1: Cho phép toán dropping $D_4^2: \mathbb{A}^\omega \to \mathbb{A}^\omega$, với $l = 4$ và $k = 2$ được cho bởi $D_4^2(\sigma) = (\sigma(0), \sigma(1), \sigma(3), \sigma(4), \sigma(5), \sigma(7), \dots)$, như vậy sẽ loại bỏ phần tử thứ 3 từ mỗi 4 phần tử mới đến và giữ lại tất cả các phần tử còn lại. Chú ý rằng luôn luôn bắt đầu đếm từ 0, do đó $\sigma(2)$, $\sigma(6)$ đã bị loại bỏ khỏi stream con σ trên. Do vậy, phép toán dropping D_4^2 đã trải qua chu kỳ 4 và kích thước khối 3 (Bảng 2.5).

Bảng 2.5. Phép toán dropping D_4^2 đã trải qua chu kỳ 4 và kích thước khối 3

0	1	2	3	4	5	6	7	8	…
$\sigma(0)$	$\sigma(1)$	~~$\sigma(2)$~~	$\sigma(3)$	$\sigma(4)$	$\sigma(5)$	~~$\sigma(6)$~~	$\sigma(7)$	$\sigma(8)$	…

Ví dụ 2.2: Cho phép toán dropping $D_4^0: \mathbb{A}^\omega \to \mathbb{A}^\omega$ với $l = 4$ và $k = 0$ được cho bởi $D_4^0(\sigma) = (\sigma(1), \sigma(2), \sigma(3), \sigma(5), \sigma(6), \sigma(7), \dots)$, như vậy sẽ loại bỏ phần tử thứ 0 từ mỗi 4 phần tử mới đến và giữ lại tất cả phần tử còn lại. Chú ý rằng luôn luôn bắt đầu đếm từ 0, do đó $\sigma(0)$, $\sigma(4)$ đã bị loại bỏ khỏi stream con σ. Do đó, phép toán loại bỏ D_4^0 đã trải qua có chu kỳ 4 và kích thước khối 3 (Bảng 2.6).

Bảng 2.6. Phép toán dropping D_4^2 đã trải qua chu kỳ 4 và kích thước khối 3

0	1	2	3	4	5	6	7	8	…
~~$\sigma(0)$~~	$\sigma(1)$	$\sigma(2)$	$\sigma(3)$	~~$\sigma(4)$~~	$\sigma(5)$	$\sigma(6)$	$\sigma(7)$	$\sigma(8)$	…

Ví dụ 2.3: Phép toán dropping có dạng $even$: $\mathbb{A}^\omega \to \mathbb{A}^\omega$ được cho bởi $even(\sigma) = (\sigma(0), \sigma(2), \sigma(4), \dots)$ lấy mỗi hai phần tử mới vào đầu tiên và bỏ qua phần tử thứ hai. Như vậy hàm $even$: $\mathbb{A}^\omega \to \mathbb{A}^\omega$ đã có chu kỳ 2 đầu vào và kích thước khối 1 đầu ra. Luôn bắt đầu đếm từ 0, do đó $\sigma(1)$, $\sigma(3), \sigma(5), \sigma(7)$ đã bị loại bỏ khỏi stream con σ của stream gốc. Ta xem mẫu stream con σ sau (Bảng 2.7):

Bảng 2.7. Hàm $even$: $\mathbb{A}^\omega \to \mathbb{A}^\omega$ đã trải qua chu kỳ 2 và kích thước khối 1

0	1	2	3	4	5	6	7	8	…
$\sigma(0)$	~~$\sigma(1)$~~	$\sigma(2)$	~~$\sigma(3)$~~	$\sigma(4)$	~~$\sigma(5)$~~	$\sigma(6)$	~~$\sigma(7)$~~	$\sigma(8)$	…

2.3.3.2. Phép toán Taking

Trong tiểu mục này, phép toán taking được giới thiệu, với phép toán này các

stream dạng BDL có thể được lấy và được định nghĩa như sau:

Định nghĩa 2.4: Với $\sigma_0, \sigma_1, \sigma_2, \cdots, \sigma_{k-1} \in \mathbb{A}^\omega$ được gọi là các stream dạng BDL và một stream σ gồm có các frame $\sigma(0)$, $\sigma(1)$, $\sigma(2)$, $\sigma(3)$, ... Cho $l \geq 2$ và $0 \leq i < l$, phép toán taking $T_l^i: \mathbb{A}^\omega \to \mathbb{A}^\omega$ dùng cho một stream $T_l^i(\sigma)$ trích dãy con theo chu kỳ l bắt đầu ở vị trí i như sau $T_l^i(\sigma) = (\sigma(i), \sigma(i+l), \sigma(i+2l), \sigma(i+3l), \dots)$ được định nghĩa bằng một phương trình vi phân stream $\left(T_l^i(\sigma)\right)' = \left(T_l^i(\sigma)\right)^l$ và $T_l^i(\sigma)(0) = \sigma(i)$ (Bảng 2.8).

Bảng 2.8. Phương trình vi phân stream của phép toán taking

Phương trình vi phân stream	Giá trị ban đầu	Điều kiện
$\left(T_l^i(\sigma)\right)' = \left(T_l^i(\sigma)\right)^l$	$T_l^i(\sigma)(0) = \sigma(i)$	$l \geq 2$ và $0 \leq i < l$

Mệnh đề 2.2: Với mọi $i \geq 0$, đặt $l = i + 2$. Khi đó $\left(T_l^i(\sigma)\right)' = \left(T_l^i(\sigma)\right)^l$.

Chứng minh: Kiểm chứng hình thức mệnh đề 2.2 bằng quy nạp theo i.

- Cơ sở quy nạp ($i = 0$, $l = 2$). Ta có

$$T_2^0(\sigma) = (\sigma(0), \sigma(2), \sigma(4), \dots).$$

Suy ra $$\left(T_2^0(\sigma)\right)' = (\sigma(2), \sigma(4), \sigma(6), \dots) = \left(T_2^0(\sigma)\right)^2.$$

Điều này đúng theo định nghĩa 2.4 của phép toán Taking.

- Giả thuyết quy nạp. Giả sử với $i = k$, đặt $l = k + 2$ ta có

$$\left(T_{k+2}^k(\sigma)\right)' = \left(T_{k+2}^k(\sigma)\right)^{k+2}.$$

- Bước quy nạp ($i = k + 1$, $l = k + 3$). Khi đó

$$T_{k+3}^{k+1}(\sigma) = \left(\sigma(k+1), \sigma\left((k+1)+(k+3)\right), \sigma\left((k+1)+2(k+3)\right), \dots\right) = (\sigma(k+1), \sigma(2k+4), \sigma(3k+7), \dots).$$

Lấy đạo hàm stream:

$$\left(T_{k+3}^{k+1}(\sigma)\right)' = (\sigma(2k+4), \sigma(3k+7), \dots).$$

Mặt khác, dịch $k + 3$ bước trên $T_{k+3}^{k+1}(\sigma)$ cũng cho:

$$\left(T_{k+3}^{k+1}(\sigma)\right)^{k+3} = (\sigma(2k+4), \sigma(3k+7), \dots).$$

Suy ra

$$\left(T_{k+3}^{k+1}(\sigma)\right)' = \left(T_{k+3}^{k+1}(\sigma)\right)^{k+3}.$$

Hoàn tất bước quy nạp.

- Theo nguyên lý quy nạp, mệnh đề 2.2 đúng với mọi $i \geq 0$, $l = i + 2$ □

Ví dụ 2.4: Cho phép toán taking có chu kỳ 3 và kích thước khối 2 như sau: T_3^2 kết quả sẽ là $T_3^2 = (\sigma(2), \sigma(5), \sigma(8), \sigma(11), \dots)$ (Bảng 2.9).

Bảng 2.9. Cho σ ban đầu và $T_3^2 = (\sigma(2), \sigma(5), \sigma(8), \cdots)$ lấy từng frame strong stream

0	1	2	3	4	5	6	7	8	…
$\sigma(0)$	$\sigma(1)$	$\sigma(2)$	$\sigma(3)$	$\sigma(4)$	$\sigma(5)$	$\sigma(6)$	$\sigma(7)$	$\sigma(8)$	…

$T_3^2 = (\sigma(2), \sigma(5), \sigma(8), \cdots)$ lấy từng frame strong stream									
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**…**
		$\sigma(2)$			$\sigma(5)$			$\sigma(8)$	…

2.3.3.3. Phép toán Zipping

Trong tiểu mục này, phép toán zipping được giới thiệu, với phép toán này các stream dạng BDL có thể được nén và được định nghĩa như sau:

Định nghĩa 2.5: Với $\sigma_0, \sigma_1, \sigma_2, \dots, \sigma_{k-1}$ được gọi là các stream dạng BDL và một stream σ gồm có $\sigma(0)$, $\sigma(1)$, $\sigma(2)$, $\sigma(3), \dots$ Cho $k \geq 1$ và các stream $\sigma_0, \sigma_1, \sigma_2, \dots, \sigma_{k-1} \in \mathbb{A}^\omega$, phép toán zipping $Z_k: (\mathbb{A}^\omega)^k \to \mathbb{A}^\omega$ được định nghĩa bằng phương trình vi phân stream $Z_k(\sigma_0, \dots, \sigma_{k-1})(0) = \sigma_0(0)$ và $Z_k(\sigma_0, \dots, \sigma_{k-1})' = Z_k(\sigma_1, \dots, \sigma_{k-1}, \sigma_0')$ (Bảng 2.10).

Bảng 2.10. Phương trình vi phân stream của phép toán zipping

Phương trình vi phân stream (Tail)	Giá trị ban đầu (Head)	Điều kiện
$Z_k(\sigma_0, \dots, \sigma_{k-1})' = Z_k(\sigma_1, \dots, \sigma_{k-1}, \sigma_0')$	$Z_k(\sigma_0, \dots, \sigma_{k-1})(0) = \sigma_0(0)$	$k \geq 1$

Ví dụ 2.5: Cho $\sigma_0, \sigma_1, \sigma_2, \dots, \sigma_{k-1}, \tau_0, \tau_1, \tau_2, \dots, \tau_{k-1} \in \mathbb{A}^\omega$ là các stream, với stream σ gồm có $\sigma(0), \sigma(1), \sigma(2), \dots$ và với stream τ gồm có $\tau(0)$, $\tau(1), \tau(2), \dots$:

- Xét phép toán zipping với $k = 2$ sau: $Z_2(\sigma, \tau) = (\sigma(0), \tau(0), \sigma(1), \tau(1), \dots)$
- Xét phép toán zipping với $k = 3$ như sau: $Z_3(\sigma_0, \sigma_1, \sigma_2) = (\sigma_0(0), Z_3(\sigma_1, \sigma_2, \sigma_0')) = (\sigma_0(0), \sigma_1(0), Z_3(\sigma_2, \sigma_0', \sigma_1')) = (\sigma_0(0), \sigma_1(0), \sigma_2(0), Z_3(\sigma_0', \sigma_1', \sigma_2')) = (\sigma_0(0), \sigma_1(0), \sigma_2(0), \sigma_0'(0), Z_3(\sigma_1', \sigma_2', \sigma_0'')) = (\sigma_0(0), \sigma_1(0), \sigma_2(0), \sigma_0'(0), \sigma_1'(0), Z_3(\sigma_2', \sigma_0'', \sigma_1'')) = (\sigma_0(0), \sigma_1(0), \sigma_2(0), \sigma_0'(0), \sigma_1'(0), \sigma_2'(0), Z_3(\sigma_0'', \sigma_1'', \sigma_2''))$.

2.3.3.4. Phép toán Splitting

Trong tiểu mục này, phép toán splitting được giới thiệu với các stream dạng BDL có thể được tách và được định nghĩa như sau:

Định nghĩa 2.6: Cho $k, l \in \mathbb{N}$ với $l > 1$ và $1 \leq k \leq l$, bất kỳ dãy số tuần tự k: $0 \leq n_0 < n_1 < \cdots < n_{k-1} < l$ đều xác định một bộ lấy mẫu stream chu kỳ $S: \mathbb{A}^\omega \to \mathbb{A}^\omega$ của chu kỳ l đầu vào và kích thước khối k đầu ra như vậy được định nghĩa bằng phương trình vi phân stream của phép toán splitting (Bảng 2.11).

Bảng 2.11. Phương trình vi phân stream của phép toán splitting

Phương trình vi phân stream	Giá trị ban đầu	Điều kiện
$S(\sigma)^{(k)} = S(\sigma^{(l)})$	$S(\sigma)(j) = \sigma(n_j)$	$0 \leq j < k, l > 1$ và $1 \leq k \leq l$
Với $S(\sigma) = Z_k\left(T_l^{n_0}(\sigma), T_l^{n_1}(\sigma), \cdots, T_l^{n_{k-1}}(\sigma)\right)$		

Ví dụ 2.6: Cho stream σ và với $k = 2$ và $l = 4$. Hãy tính $S(\sigma)$.

- $S(\sigma) = Z_2\left(T_4^0(\sigma), T_4^1(\sigma)\right) = Z_2\left((\sigma(0), \sigma(4), \dots), (\sigma(1), \sigma(5), \dots)\right) = (\sigma(0), \sigma(1), \sigma(4), \sigma(5), \dots)$
- $S(\sigma)^{(2)} = (\sigma(0), \sigma(1), \sigma(4), \sigma(5), \dots)$
- $S(\sigma^{(4)}) = Z_2\left(T_4^0(\sigma), T_4^1(\sigma)\right) = Z_2\left((\sigma(0), \sigma(4), \dots), (\sigma(1), \sigma(5), \dots)\right) =$

$(\sigma(0), \sigma(1), \sigma(4), \sigma(5), \dots)$

- $S(\sigma)^{(k)} = S(\sigma^{(l)}) \rightarrow S(\sigma)^{(2)} = S(\sigma^{(4)})$

2.3.3.5. Phép toán Reversing

Các phép chuyển đổi stream chu kỳ với phép toán taking và zipping đã được định nghĩa ở trên, không chỉ tạo ra các dãy con mà còn có thể thay đổi thứ tự các stream dạng BDL. Phép toán reversing stream được định nghĩa như sau:

Định nghĩa 2.7: Cho phép toán reversing Rev_k được định nghĩa bằng hàm sau: $Rev_k: \mathbb{A}^\omega \rightarrow \mathbb{A}^\omega$, với $k \geq 1$ được tổ hợp từ phép toán zipping và taking thành biểu thức stream như sau: $Rev_k(\sigma) = Z_k\left(T_k^{k-1}(\sigma), T_k^{k-2}(\sigma), \cdots, T_k^0(\sigma)\right)$ (Bảng 2.12).

Các phép toán biến đổi stream chu kỳ được định nghĩa sao cho không chỉ tạo ra các đoạn stream con mà còn có thể thay đổi thứ tự các stream trong đoạn con đó. Phép toán reversing dạng $Rev_k: \mathbb{A}^\omega \rightarrow \mathbb{A}^\omega$ để lật ngược một stream với $k \geq 1$. Cụ thể, $Rev_k(\sigma)$ ghép nối k đoạn con của stream σ theo thứ tự ngược lại.

Bảng 2.12. Phương trình vi phân stream của phép toán reversing

Phương trình vi phân stream (Tail)	Giá trị ban đầu (Head)	Điều kiện
$Rev_k(\sigma)' = Rev_k(\sigma'')$	$Rev_k(\sigma)(k) = \sigma(k)$	$k \geq 1$
Với $Rev_k(\sigma) = Z_k\left(T_k^{k-1}(\sigma), T_k^{k-2}(\sigma), \cdots, T_k^0(\sigma)\right)$		

Ví dụ 2.7: Cho stream $\sigma = \sigma(0), \sigma(1), \sigma(2), \sigma(3), \sigma(4), \dots$ và với $k = 3$. Hãy sử dụng phép toán reversing để tính stream σ đảo ngược.

- $Rev_3(\sigma) = Z_3\left(T_3^2(\sigma), T_3^1(\sigma), T_3^0(\sigma)\right) = Z_3\left(\sigma(2), \sigma(1), \sigma(0)\right)$
- $Rev_3(\sigma)' = Z_3(T_3^2(\sigma)', T_3^1(\sigma)', T_3^0(\sigma)') = Z_3\left(\sigma(4), \sigma(3), \sigma(2)\right)$
- $Rev_3(\sigma'') = Z_3\left(T_3^2(\sigma''), T_3^1(\sigma''), T_3^0(\sigma'')\right) = Z_3\left(\sigma(5), \sigma(4), \sigma(3)\right)$

Giả sử $k = 3$, ta có phép toán reverse $Rev_3(\sigma)$ là stream của các frame được tạo ra bằng cách lật ngược các frame của stream trong các đoạn con đó $\sigma(0), \sigma(1), \sigma(2), \sigma(3), \sigma(4), \sigma(5), \sigma(6), \sigma(7), \sigma(8), \dots$ theo thứ tự lật ngược tương ứng với $k = 3$ sẽ là $\sigma(2), \sigma(1), \sigma(0), \sigma(5), \sigma(4), \sigma(3), \sigma(8), \sigma(7), \sigma(6), \dots$

2.3.3.6. Phép toán Merging

Trong tiểu mục này, phép toán merging được giới thiệu với các stream dạng BDL có thể được trộn và được định nghĩa như sau:

Định nghĩa 2.8: Cho hai stream $\sigma = (\sigma(0), \sigma(1), \sigma(2), \dots) \in \mathbb{A}^\omega$ và $\tau = (\tau(0), \tau(1), \tau(2), \dots) \in \mathbb{A}^\omega$, áp dụng phép toán merging hai stream σ và τ ta được $\sigma + \tau = (\sigma(0) + \tau(0), \sigma(1) + \tau(1), \sigma(2) + \tau(2), \sigma(3) + \tau(3), \dots)$ hoặc biểu diễn bằng giá trị ban đầu và phương trình vi phân stream sau (Bảng 2.13).

Bảng 2.13. Phương trình vi phân stream của phép toán merging

Phương trình vi phân stream	Giá trị ban đầu	Điều kiện
$(\sigma + \tau)' = \sigma' + \tau'$	$(\sigma + \tau)(0) = \sigma(0) + \tau(0)$	$\sigma(0), \dots, \tau(0), \dots \in \mathbb{A}^\omega$

Trong đó, phép toán merging được ký hiệu $+$, $\sigma + \tau$ được ký hiệu là gộp hai

stream, còn $\sigma(i) + \tau(i)$ được ký hiệu là gộp hai frame thứ i. Định nghĩa này cho phép gộp frame r với một stream σ ta có được $[r] + \sigma = (r, 0, 0, 0, \dots) + (\sigma(0), \sigma(1), \sigma(2), \dots) = (r + \sigma(0), \sigma(1), \sigma(2), \dots)$. Phép toán merging $r + \sigma$ được sử dụng để chỉ $[r] + \sigma$, tùy theo ngữ cảnh sẽ làm rõ ký hiệu r là một frame r hoặc $[r]$ là một stream $[r]$.

Tính chất 2.1: $\forall r, s \in \mathbb{A}$ và $\sigma, \tau, \rho \in \mathbb{A}^{\omega}$, ta có một số tính chất của phép toán merging stream như sau:

- $[r] + [s] = [r + s]$ (2.4)
- $\sigma + 0 = \sigma$ (2.5)
- $\sigma + \tau = \tau + \sigma$ (2.6)
- $\sigma + (\tau + \rho) = (\sigma + \tau) + \rho$ (2.7)

Ví dụ 2.8: Cho $\sigma = \sigma(0), \sigma(1), \dots$ là stream dữ liệu văn bản và video, và $\tau = \tau(0), \tau(1), \dots$ là stream dữ liệu văn bản và âm thanh, sử dụng phép toán merging ta được một stream dạng BDL như sau: $\mu = \sigma + \tau = (\sigma(0) + \tau(0), \sigma(1) + \tau(1), \sigma(2) + \tau(2), \dots)$. Trong đó, $\sigma(i) = \big(\text{text}_{\sigma}(i), \text{video}(i)\big)$, $\tau(i) = \big(\text{text}_{\tau}(i), \text{audio}(i)\big)$ là từng frame gồm các thành phần dữ liệu được gộp cùng thời điểm vào một frame mới là $(\sigma + \tau)(i) = \sigma(i) + \tau(i) = \big(\text{text}_{\sigma}(i) + \text{text}_{\tau}(i), \text{video}(i), \text{audio}(i)\big)$. Lúc này, đưa dữ liệu vào từng frame như sau: $\sigma(0) = (\{\text{"Xe máy"}\}, \text{"vd\_xm.mp4"})$, $\sigma(1) = (\{\text{"Người đi bộ"}\}, \text{"vd\_ndb.mp4"})$, $\tau(0) = (\{\text{"Tiếng còi"}\}, \text{"au\_xm.wav"})$, và $\tau(1) = (\{\text{"Tiếng bước chân"}\}, \text{"au\_tbc.wav"})$. Do vậy, kết quả gộp các frame của hai stream σ và τ sẽ là: $(\sigma + \tau)(0) = (\{\text{"Xe máy"}, \text{"Tiếng còi"}\}, \text{"vd\_xm.mp4"}, \text{"au\_xm.wav"})$ và $(\sigma + \tau)(1) = (\{\text{"Người đi bộ"}, \text{"Tiếng bước chân"}\}, \text{"vd\_ndb.mp4"}, \text{"au\_tbc.wav"})$.

2.3.3.7. Phép toán (Convolution) product

Trong tiểu mục này, phép toán (convolution) product ($\times$) được định nghĩa sau:

Định nghĩa 2.9: Cho hai stream σ và $\tau \in \mathbb{A}^{\omega}$, phép toán (convolution) product định nghĩa về tích chập của hai stream như sau: $\sigma \times \tau = \big(\sigma(0) + \tau(0), \big(\sigma(0) + \tau(1)\big) + \big(\sigma(1) + \tau(0)\big), \big(\sigma(0) + \tau(2)\big) + \big(\sigma(1) + \tau(1)\big) + \big(\sigma(2) + \tau(0)\big), \big(\sigma(0) + \tau(3)\big) + \big(\sigma(1) + \tau(2)\big) + \big(\sigma(2) + \tau(1)\big) + \big(\sigma(3) + \tau(0)\big), \dots\big)$. Với $0 \le i \le n$, ta có phương trình vi phân stream (Bảng 2.14) và phương trình tích chập cho hai stream (2.8) như sau:

$$(\sigma \times \tau)(n) = \sum_{i=0}^{n} \big(\sigma(i) + \tau(n - i)\big) \quad (2.8)$$

Trong đó:

- $\sigma \times \tau$ hoặc $\sigma\tau$ được ký hiệu là một tích chập của hai stream σ và $\tau \in \mathbb{A}^{\omega}$.
- $\sigma(i) + \tau(j)$ được ký hiệu là một phép toán merging của hai frame $\mathbb{A}$.
- $\sigma^0 = 1$ và $\sigma^{n+1} = \sigma \times \sigma^n$ được ký hiệu là một tích chập của các stream.
- Cho $k \in \mathbb{N}, k \times \sigma = \sum_{i=0}^{k} \sigma$ được ký hiệu là một phép toán merging của các stream.

Bảng 2.14. Phương trình vi phân stream của phép toán (convolution) product

Phương trình vi phân stream	Giá trị ban đầu	Điều kiện
$(\sigma \times \tau)' = (\sigma' \times \tau) + (\sigma(0) \times \tau')$	$(\sigma \times \tau)(0) = \sigma(0) \times \tau(0)$	$0 \le i \le n$

Phép toán (Convolution) product trong Định nghĩa 2.9, luận án phân biệt trường hợp hữu hạn và vô hạn đối với độ dài của các stream thành phần. Cụ thể, nếu hai stream σ và τ có độ dài hữu hạn, giả sử σ có độ dài m và τ có độ dài k, thì độ dài của tích chập $\sigma \times \tau$ được xác định là $m + k - 1$. Khi đó, miền giá trị của chỉ số n trong biểu thức tổng quát sẽ là $0 \le n \le m + k - 2$. Trong trường hợp các stream là vô hạn, phép tích chập luôn xác định với mọi $n \in \mathbb{N}$. Việc nêu rõ giả thiết về độ dài stream giúp đảm bảo tính chặt chẽ và tính khả thi của phép toán trong các bối cảnh ứng dụng khác nhau.

Phép toán (convolution) product tính tích chập ở trên cho phép tính tích các frame r với một stream σ, phương trình tích chập các frame với một stream $[r] \times \sigma = (r, 0, 0, 0, 0, \dots) \times (\sigma(0), \sigma(1), \sigma(2), \sigma(3), \dots) = (r + \sigma(0),\ r + \sigma(1), r + \sigma(2), r + \sigma(3), \dots)$. Kiểu hình thức $r \times \sigma$ được sử dụng để ký hiệu $[r] \times \sigma$ và cơ chế được sử dụng như sau: $-\sigma = -1 \times \sigma = (-1) \times \sigma = (-\sigma(0), -\sigma(1), -\sigma(2), -\sigma(3), \dots)$. Dấu " $-$ " này biểu diễn một stream có tất cả frame là đối của stream σ. Cụ thể, khi viết $(-1) \times \sigma$, ta đang lấy stream hằng $(-1, 0, 0, 0, \dots)$ *(Tiểu mục 1.6.7, Chương 1)* và thực hiện phép toán (convolution) product với một stream $\sigma = (\sigma(0), \sigma(1), \sigma(2), \sigma(3), \dots)$. Kết quả sẽ là $(-1) \times \sigma = (-\sigma(0), -\sigma(1), -\sigma(2), -\sigma(3),\ \dots)$, xem bảng 2.15 tính tích chập $(-1) \times \sigma$ sau đây:

Bảng 2.15. Phép toán (convolution) product tính tích chập của $(-1) \times \sigma$

Chỉ số n	Công thức frame tích chập	Kết quả frame $(-1) \times \sigma$
0	$(-1)\ \times \sigma(0)$	$-\sigma(0)$
1	$(-1)\ \times \sigma(1) + 0\ \times \sigma(0)$	$-\sigma(1)$
2	$(-1)\ \times \sigma(2) + 0\ \times \sigma(1) + 0\ \times \sigma(0)$	$-\sigma(2)$
3	$(-1)\ \times \sigma(3) + 0\ \times \sigma(2) + 0\ \times \sigma(1) + 0\ \times \sigma(0)$	$-\sigma(3)$

Tính chất 2.2: Với mọi $r, s \in \mathbb{R}$ và $\sigma, \tau, \rho \in \mathbb{A}^{\omega}$, ta có các tính chất sau:

- $[r] \times [s] = [r + s]$ (2.9)
- $0 \times \sigma = 0$ (2.10)
- $1 \times \sigma = \sigma$ (2.11)
- $\sigma \times \tau = \tau \times \sigma$ (2.12)
- $\sigma \times (\tau + \rho) = (\sigma \times \tau) + (\tau \times \sigma)$ (2.13)
- $\sigma \times (\tau \times \rho) = (\sigma \times \tau) \times \rho.$ (2.14)

Để diễn đạt sự thỏa mãn khi biểu diễn stream dạng $\sigma = (r_0, r_1, r_2, \dots, r_n, 0, 0,\ \dots)$ với $n \ge 0$ và $r_0, r_1, r_2, \dots, r_n \in \mathbb{R}$, hằng số quan trọng được sử dụng bằng biểu thức: $X = (0, 1, 0, 0,\ \dots)$ *(Tiểu mục 1.6.7, (1.3), Chương 1)* hoặc biểu diễn giá trị đầu và phương trình vi phân stream ở bảng 2.16.

Bảng 2.16. Biểu diễn giá trị ban đầu và phương trình vi phân stream

Phương trình vi phân stream (Tail)	Giá trị ban đầu (Head)	Điều kiện
$X(0)' = \sigma(1)$	$X(0) = \sigma(0)$	$n \geq 0$

Thực tế, bằng cách sử dụng biến hình thức *X (Bảng 1.12 (Stt: 3), Tiểu mục 1.6.7, Chương 1)*, các phép toán xử lý stream được mô tả một cách rõ ràng và đầy đủ như sau:

- $X \times r = (0, r, 0, 0, \dots)$ (2.15)
- $X \times \sigma = (0, \sigma(0), \sigma(1), \sigma(2), \sigma(3), \dots)$ (2.16)
- $X^n = (0, \dots_{n-2 \text{ lần}} \dots, 0, 1, 0, 0, \dots)$ (2.17)
- $r_0 + r_1 X + r_2 X^2 + \dots + r_n X^n = (r_0, r_1, \dots, r_n, 0, 0, \dots)$ (2.18)

Từ phép toán stream số (2.15) đến phép toán stream số (2.17) ở trên, suy ra rằng stream cuối cùng là phép toán stream số (2.18) đã được hình thành và được biểu diễn thành một stream đa thức mới. Một biểu thức các stream duy nhất chỉ bao gồm các stream hằng được biểu diễn như một biểu thức dạng đóng.

Ví dụ 2.9: Cho $\sigma = \sigma(0), \sigma(1), \dots$ là stream dữ liệu văn bản và video, và $\tau = \tau(0), \tau(1), \dots$ là stream dữ liệu văn bản và âm thanh, sử dụng phép toán (convolution) product ta tính được tích chập của một stream dạng BDL như sau: $\sigma \times \tau = \big(\sigma(0) + \tau(0), \big(\sigma(0) + \tau(1)\big) + \big(\sigma(1) + \tau(0)\big), \big(\sigma(0) + \tau(2)\big) + \big(\sigma(1) + \tau(1)\big) + \big(\sigma(2) + \tau(0)\big), \big(\sigma(0) + \tau(3)\big) + \big(\sigma(1) + \tau(2)\big) + \big(\sigma(2) + \tau(1)\big) + \big(\sigma(3) + \tau(0)\big), \dots\big)$:

- Giả sử cho 4 frame của hai stream σ và τ dữ liệu dạng BDL như sau:
  - $\sigma(0) = (\{$"Xe máy", "au_xm. wav", "vd_xm. mp4"$\})$,
  - $\sigma(1) = (\{$"Người đi bộ", "au_ndb. wav", "vd_ndb. mp4"$\})$,
  - $\tau(0) = (\{$"Tiếng còi", "au_tc. wav", "vd_tc. mp4"$\})$, và
  - $\tau(1) = (\{$"Tiếng bước chân", "au_tbc. wav", "vd_tbc. mp4"$\})$.
- Tính từng frame của tích chập $\sigma \times \tau$ như sau:
  - $(\sigma \times \tau)(0) = \big(\sigma(0) + \tau(0)\big) = \begin{pmatrix} \{\text{"Xe máy", "Tiếng còi"}\}, \\ \{\text{"au_xm.wav", "au_tc.wav"}\}, \\ \{\text{"vd_xm. mp4", "vd_tc. mp4"}\} \end{pmatrix}$
  - $(\sigma \times \tau)(1) = \big(\sigma(0) + \tau(1)\big) + \big(\sigma(1) + \tau(0)\big) =$
$$\begin{pmatrix} \{\text{"Xe máy", "Tiếng bước chân"}\}, \\ \{\text{"au_xm.wav", "au_tbc.wav"}\}, \\ \{\text{"vd_xm. mp4", "vd_tbc. mp4"}\} \end{pmatrix} + \begin{pmatrix} \{\text{"Người đi bộ", "Tiếng còi"}\}, \\ \{\text{"au_ndb.wav", "au_tc.wav"}\}, \\ \{\text{"vd_ndb. mp4", "vd_tc. mp4"}\} \end{pmatrix} =$$
$$\begin{pmatrix} \{\text{"Xe máy", "Tiếng bước chân", "Người đi bộ", "Tiếng còi"}\}, \\ \{\text{"au_xm. wav", "au_tbc. wav", "au_ndb. wav", "au_tc. wav"}\}, \\ \{\text{"vd_xm. mp4", "vd_tbc. mp4", "vd_ndb. mp4", "vd_tc. mp4"}\} \end{pmatrix}$$

Ví dụ 2.9 mô tả phép toán (convolution) product trên stream dạng BDL đã tổng hợp nội dung theo thời gian như sau: Trộn thông tin mô tả từ hai stream (Text), trộn âm thanh từ hai stream (Audio), và hiển thị kết hợp hai stream (Video). Việc trộn âm thanh về nguyên tắc tương ứng với phép cộng hoặc ghép chồng tín hiệu âm thanh. Nếu thực hiện trực tiếp, điều này có thể dẫn đến hiện tượng tạp âm do sự cộng hưởng hoặc xung đột tín hiệu. Để hạn chế vấn đề này, trong ứng dụng thực tế cần áp dụng các kỹ thuật chuẩn hóa âm lượng, lọc tín hiệu hoặc chọn chiến lược trộn thích hợp (overlay-chồng hình, ưu tiên

stream chính/phụ). Đối với việc hiển thị kết hợp video, có thể lựa chọn nhiều cơ chế khác nhau như chồng hình, hiển thị dạng cửa sổ lồng trong cửa sổ, hiển thị song song hoặc kết hợp ở mức frame. Trong phạm vi BDL, phép tích chập được hiểu như sự hợp nhất ngữ nghĩa để tạo ra frame đa phương tiện mới, còn cơ chế hiển thị cụ thể sẽ phụ thuộc vào mục tiêu ứng dụng.

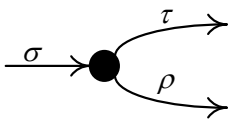
2.3.3.8. Phép toán Coppying

Trong tiểu mục này, phép toán coppying (•) được định nghĩa như sau:

Định nghĩa 2.10: Cho $\sigma, \tau, \rho \in \mathbb{A}^{\omega}$ và $n \geq 0$, phép toán copying nhận giá trị của stream là frame $\sigma(n)$ và phép toán này xuất ra hai bản sao của stream là frame giống nhau $\tau(n)$ và $\rho(n)$ tuân theo biểu thức sau: $\sigma(n) = \tau(n) = \rho(n), \forall n \geq 0$ hoặc $\sigma = \tau = \rho$, hoặc biểu diễn giá trị đầu và phương trình vi phân stream như sau (Bảng 2.17, Hình 2.5).

Bảng 2.17. Phương trình vi phân stream của phép toán coppying

Phương trình vi phân stream	Giá trị ban đầu	Điều kiện
$\sigma' = \tau' = \rho'$	$\sigma(0) = \tau(0) = \rho(0)$	$n \geq 0$



Hình 2.5. Phép toán copying dùng cho ba stream σ, τ và ρ

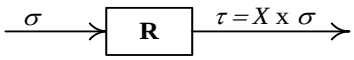
2.3.3.9. Phép toán Registering

Trong tiểu mục này, phép toán registering (■) được định nghĩa như sau:

Định nghĩa 2.11: Cho $\sigma = (\sigma(0), \sigma(1), \dots) \in \mathbb{A}^{\omega}$ và $\tau = (\tau(0), \tau(1), \dots) \in \mathbb{A}^{\omega}$ là hai stream, một phép toán registering giữa hai σ và τ được định nghĩa: $\tau = X \times \sigma = (0, \sigma(0), \sigma(1), \sigma(2), \sigma(3), \sigma(4), \sigma(5), \dots)$ hoặc biểu diễn giá trị ban đầu và phương trình vi phân stream như sau (Bảng 2.18).

Bảng 2.18. Phương trình vi phân stream của phép toán registering

Phương trình vi phân stream	Giá trị ban đầu
$(\tau)' = \sigma$	$\tau(0) = \sigma(0)$



Hình 2.6. Phép toán registering dùng cho stream σ và τ

Phép toán registering được mô tả trực quan ở hình 2.6. Có thể quan sát thấy rằng phép toán này bao gồm một ô nhớ nơi có chứa stream ban đầu $\sigma(0)$. Phép toán này bắt đầu hoạt động của mình, tại thời điểm 0, bằng cách đưa ra giá trị stream $\tau(0) = \sigma(0)$, trong khi đồng thời nhập stream $\sigma(1)$ và lưu trữ nó vào trong ô nhớ. Tại mọi thời điểm $n \geq 1$, stream $\tau(n) = \sigma(n-1)$ được đưa ra và stream $\sigma(n)$ được nhập vào và được lưu trữ. Phép toán này đôi khi cũng được gọi là phép toán độ trễ đơn vị.

2.3.3.10. Phép toán Assignment

Trong tiểu mục này, phép toán assignment (:=) được định nghĩa như sau:

Định nghĩa 2.12: Cho $\sigma = (\sigma(0), \sigma(1), \dots) \in \mathbb{A}^{\omega}$ và $\tau = (\tau(0), \tau(1), \dots) \in \mathbb{A}^{\omega}$ là hai stream, việc gán một stream σ cho stream τ được định nghĩa như sau $\tau := \sigma$. Được hiểu là tại mọi thời điểm $i \geq 0$, nó thiết lập frame $\sigma(i)$ vào $\tau(i)$ được hiểu là $\tau(i) := \sigma(i)$ hoặc biểu diễn giá trị ban đầu và phương trình vi phân stream như sau (bảng 2.19).

Bảng 2.19. Phương trình vi phân stream của phép toán assignment

Phương trình vi phân stream	Giá trị ban đầu
$(\tau := \sigma)' = \tau' := \sigma'$	$(\tau := \sigma)(0) = \tau(0) := \sigma(0)$

Tính chất 2.3: Cho tất cả các biến thuộc stream $\tau \in \mathbb{A}^{\omega}$, chúng ta có các tính chất sau.

- $\tau := 0 = 0$ (2.19)
- $\tau := 1 = 1$ (2.20)
- $\tau := \sigma = \sigma$ (2.21)

Tính chất (2.19) nghĩa là sau phép toán assignment $\tau := 0$, giá trị của stream τ là stream $[0] = (0, 0, 0, \dots)$. Tính chất (2.20) giống với tính chất (2.19) khi sử dụng giá trị 1 thay vì sử dụng giá trị 0. Tính chất (2.21) có nghĩa là sau phép toán assignment $\tau := \sigma$, giá trị của stream τ là stream $\sigma = (\sigma(0), \sigma(1), \sigma(2), \dots)$. Tính chất (2.21) là giống với tính chất (2.19) khi sử dụng giá trị $(\sigma(0), \sigma(1), \dots)$ thay vì sử dụng giá trị 0.

2.3.3.11. Những lợi ích cụ thể của các phép toán xử lý stream

Các phép toán xử lý stream từ tiểu mục 2.3.3.1 đến 2.3.3.10 được luận án xây dựng mang lại những lợi ích cụ thể như sau [CT.1][CT.2][CT.8] (Bảng 2.20):

Bảng 2.20. Những lợi ích cụ thể của các phép toán xử lý stream

Phép toán	Tiêu chí	Những lợi ích cụ thể
Dropping (D_l^k)	Tiền xử lý stream dữ liệu	Loại bỏ các phần tử đầu/cuối hoặc bỏ qua theo chu kỳ. Giúp làm sạch dữ liệu, giảm nhiễu và giảm tải trước khi đưa vào pipeline.
Taking (T_l^i)		Lấy ra l frame tiếp theo sau khi bỏ qua i frame đầu. Hữu ích trong việc trích chọn các đoạn frame, xây dựng các stream con và thực hiện thao tác lấy mẫu trong xử lý stream.
Splitting (S)	Biểu diễn các stream dữ liệu	Chia stream thành các phần nhỏ để xử lý song song, nối tiếp hoặc theo cơ chế hồi tiếp (Đưa một phần kết quả xử lý quay trở lại làm đầu vào cho chính stream đó). Hỗ trợ biểu diễn cấu trúc pipeline và tăng tính mô đun hóa (Chia hệ thống thành các mô đun nhỏ, độc lập, mỗi mô đun đảm nhiệm một chức năng rõ ràng).
Zipping (Z_k)	Việc tách, nén và hợp nhất stream Đảm bảo tính chất bảo toàn stream	Nén các stream theo từng vị trí tương ứng để tạo stream đồng bộ. Hỗ trợ hợp nhất dữ liệu từ nhiều pipeline xử lý. Đồng bộ hóa từng nhóm k-frame khi hai stream có tốc độ khác nhau. Duy trì bất biến (Được giữ ổn định trong suốt quá trình xử lý stream, bất kể các phép toán trung gian) của hệ thống tính toán và ổn định đầu vào pipeline.
Merging ($+$)	Kiểm tra tính đúng đắn và hợp nhất dữ liệu	Trộn nhiều stream thành một stream duy nhất, đảm bảo không mất dữ liệu và duy trì thứ tự. Có ích trong hợp nhất nhánh xử lý và bảo toàn tính toàn vẹn dữ liệu (Không mất dữ liệu nào từ hai stream đầu vào).
Reversing (Rev_k)	Phân tích stream và tối	Đảo ngược stream để kiểm tra tính đối xứng, tìm mẫu đảo thời gian hoặc đánh giá tính chất logic thời gian (Dùng để mô hình hóa

Phép toán	Tiêu chí	Những lợi ích cụ thể
	ưu hóa thuật toán	sự liên tục của dữ liệu, tính ổn định, tính bất biến, và hành vi tuần tự theo thời gian). Rất hữu ích trong phân tích sâu các chuỗi dữ liệu.
Registering (■)		Lưu trạng thái tạm thời (Delay) trong quá trình xử lý stream theo cơ chế đường ống để hỗ trợ các thuật toán tuần tự, phân tích từng bước hoặc thuật toán có độ trễ.
Convolution Product (×)	Xử lý tuần tự	Áp dụng tích chập lên stream để làm mượt, lọc nhiễu, trích đặc trưng hoặc khôi phục tín hiệu. Hữu ích với dữ liệu cảm biến, âm thanh, văn bản, hình ảnh.
Assignment (:=)	Hậu xử lý	Gán giá trị mới cho một stream nhằm cập nhật trạng thái hoặc tái định nghĩa stream trong các bước xử lý sau.
Copying (•)		Nhân bản một stream thành hai hoặc nhiều bản sao dùng cho các nhánh xử lý khác nhau mà không ảnh hưởng đến stream gốc.

2.3.4. Tổ hợp các phép toán stream thành các biểu thức stream dạng BDL

Tất cả các bộ lấy mẫu stream chu kỳ và tổng quát hơn với nhiều bộ chuyển đổi stream chu kỳ không nhất định phải duy trì thứ tự các frame trong một stream dạng BDL, có thể thu được bằng cách tách hoặc gộp các stream. Các phép toán stream: taking, zipping, dropping và reversing được tổ hợp, với các phép toán stream này thì các stream có thể được lấy, nén, loại bỏ và đảo ngược.

Định nghĩa 2.13: Việc kết hợp của phép toán taking và zipping trong phân tích stream dạng BDL cho các biểu thức stream như sau:

- Cho $l \geq 2$ và $0 \leq i < l$, phép toán taking có dạng $T_l^i: \mathbb{A}^\omega \to \mathbb{A}^\omega$ được định nghĩa bằng một phương trình sau: $T_l^i(\sigma)' = T_l^i(\sigma)^l, T_l^i(\sigma)(0) = \sigma(i)$.
- Cho $k \geq 2$ và dãy $\sigma_0, \ldots, \sigma_{k-1} \in \mathbb{A}^\omega$ là các stream dạng BDL (không phải là các frame), một biểu diễn stream σ có các frame sau: $\sigma(0)$, $\sigma(1)$, $\sigma(2)$,…, phép toán zipping có dạng $Z_k: (\mathbb{A}^\omega)^k \to \mathbb{A}^\omega$ được định nghĩa bằng phương trình có dạng như sau: $Z_k(\sigma_0, \ldots, \sigma_{k-1})(0) = \sigma_0(0)$, $Z_k(\sigma_0, \ldots, \sigma_{k-1})' = Z_k(\sigma_1, \ldots, \sigma_{k-1}, \sigma_0')$.
- Theo quy nạp, nếu $k \geq 2$ và $0 \leq r \leq k-1$ thì $Z_k(\sigma_0, \cdots, \sigma_{k-1})(kn + r) = \sigma_r(n)$. Phép toán zipping: $T_3^2 = (\sigma(2), \sigma(5), \sigma(8), \sigma(11), \ldots)$ và $Z_2(\sigma, \tau) = (\sigma(0), \tau(0), \sigma(1), \tau(1), \sigma(2), \tau(2))$.
- $Z_k\big(Z_l(\sigma_0, \sigma_k, \ldots, \sigma_{lk-k}), Z_l(\sigma_1, \sigma_{k+1}, \ldots, \sigma_{lk-k+1}), \ldots, Z_l(\sigma_{k-1}, \sigma_{2k-1}, \ldots, \sigma_{lk-1})\big) = Z_{lk}(\sigma_0, \sigma_1, \ldots, \sigma_{lk-1})$ cho phép tái tạo lại phép toán zipping tùy ý.

Ví dụ 2.10: Biểu thức stream $Z_k(\sigma_0, \cdots, \sigma_{k-1})(kn + r) = \sigma_r(n)$. Giả sử cho $r = 2, k = 3, n = 0$ theo đồng quy nạp, giả sử $k \geq 2$ và $0 \leq r \leq k-1$ khi đó $Z_3(\sigma_0, \sigma_1, \sigma_2)(2) = \sigma_2(0)$. Cho một trường hợp khác, giả sử rằng cho $r = 1, k = 3, n = 1$ theo đồng quy nạp, suy ra nếu $k \geq 2$ và $0 \leq r \leq k-1$ thì khi đó $Z_3(\sigma_0, \sigma_1, \sigma_2)(4) = \sigma_1(1)$.

Ví dụ 2.11: Biểu thức $Z_k(Z_l(\sigma_0, \sigma_k, \ldots, \sigma_{lk-k}), Z_l(\sigma_1, \sigma_{k+1}, \ldots, \sigma_{lk-k+1}), \ldots, Z_l(\sigma_{k-1}, \sigma_{2k-1}, \ldots, \sigma_{lk-1})) = Z_{lk}(\sigma_0, \sigma_1, \ldots, \sigma_{lk-1})$ stream. Giả sử cho $k = 2$, $l = 2$, và stream $\sigma = (\sigma(0), \sigma(1), \sigma(2), \sigma(3), \sigma(4), \ldots)$. Khi đó σ_i là stream σ dịch phải i bước $\sigma_i = (\sigma(i), \sigma(i+1), \sigma(i+2), \ldots)$. Lúc này, vế trái

$Z_2(\sigma_0,\sigma_2) = \big((\sigma(0),\sigma(2)),(\sigma(1),\sigma(3)),(\sigma(2),\sigma(4)),\dots\big)$, $Z_2(\sigma_1,\sigma_3) = \big((\sigma(1),\sigma(3)),(\sigma(2),\sigma(4)),(\sigma(3),\sigma(5)),\dots\big)$, tiếp đó $Z_2\big(Z_2(\sigma_0,\sigma_2),Z_2(\sigma_1,\sigma_3)\big) = \Big(\big((\sigma(0),\sigma(2)),(\sigma(1),\sigma(3))\big),\big((\sigma(1),\sigma(3)),(\sigma(2),\sigma(4))\big),\big((\sigma(2),\sigma(4)),(\sigma(3),\sigma(5))\big),\dots\Big)$, ghép các frame theo thứ tự ta được $\big((\sigma(0),\sigma(1),\sigma(2),\sigma(3)),(\sigma(1),\sigma(2),\sigma(3),\sigma(4)),(\sigma(2),\sigma(3),\sigma(4),\sigma(5)),\dots\big)$. Còn về phải $Z_4(\sigma_0,\sigma_1,\sigma_2,\sigma_3) = \big((\sigma(0),\sigma(1),\sigma(2),\sigma(3)),(\sigma(1),\sigma(2),\sigma(3),\sigma(4)),(\sigma(2),\sigma(3),\sigma(4),\sigma(5)),\dots\big)$. Hai về trùng nhau, nên với $k = 2$ và $l = 2$ thì $Z_2\big(Z_2(\sigma_0,\sigma_2),Z_2(\sigma_1,\sigma_3)\big) \equiv Z_4(\sigma_0,\sigma_1,\sigma_2,\sigma_3)$.

Các phân tích trên cho thấy, với bất kỳ bộ lấy mẫu stream chu kỳ nào cũng đều có thể được biểu đạt bằng các khía cạnh phép toán taking và zipping. Bộ lấy mẫu stream chu kỳ S như trong định nghĩa 2.2 và dựa vào định nghĩa 2.7 cho ta cách tổ hợp phép toán taking và zipping thành biểu thức stream dạng BDL như sau: $S(\sigma) = Z_k\left(T_l^{n_0}(\sigma), T_l^{n_1}(\sigma), \dots, T_l^{n_{k-1}}(\sigma)\right)$.

Ví dụ 2.12: Cho $k = 2$ và $l = 2$ với lựa chọn chỉ số $\mathrm{n}_0 = 0$, $\mathrm{n}_1 = 1$ (vì $0 \le \mathrm{n}_0 < \mathrm{n}_1 < 2$, dựa theo định nghĩa 2.2). Gọi $\sigma = (\sigma(0),\sigma(1),\sigma(2),\sigma(3),\sigma(4),\dots)$. Khi đó, $T_2^0(\sigma) = (\sigma(0),\sigma(2),\sigma(4),\sigma(6),\dots)$ lấy mẫu tuần hoàn bước 2 bắt đầu từ vị trí 0. $T_2^1(\sigma) = (\sigma(1),\sigma(3),\sigma(5),\sigma(7),\dots)$ lấy mẫu tuần hoàn bước 2 bắt đầu từ vị trí 1. Vậy biểu thức stream $S(\sigma) = Z_2\left(T_2^0(\sigma), T_2^1(\sigma)\right) = \big((\sigma(0),\sigma(1)),(\sigma(2),\sigma(3)),(\sigma(4),\sigma(5)),(\sigma(6),\sigma(7)),\dots\big)$.

Tổng quát hơn, các phép chuyển đổi stream chu kỳ với phép toán taking và zipping được định nghĩa trên stream dạng BDL, không chỉ tạo ra các dãy stream con mà còn có thể thay đổi thứ tự của các phần tử stream con đó. Đối với trường hợp cụ thể, phép toán reverse các stream Rev_k được mô tả bằng hàm đảo ngược các stream dạng BDL như sau: $Rev_k\colon \mathbb{A}^\omega \to \mathbb{A}^\omega$, trong đó với $k \ge 1$ được dùng cho hàm đảo ngược các stream bằng việc tổ hợp phép toán zipping và taking thành biểu thức stream đảo ngược như sau: $Rev_k(\sigma) = Z_k\left(T_k^{k-1}(\sigma), T_k^{k-2}(\sigma), \cdots, T_k^0(\sigma)\right)$.

Ví dụ 2.13: Cho phép đảo ngược Rev với $k = 2$, và $\sigma = (\sigma(0),\sigma(1),\sigma(2),\sigma(3),\sigma(4),\dots)$. Khi đó, $T_2^0(\sigma) = (\sigma(0),\sigma(2),\sigma(4),\sigma(6),\dots)$ (Lấy frame ở vị trí chẵn). $T_2^1(\sigma) = (\sigma(1),\sigma(3),\sigma(5),\sigma(7),\dots)$ (Lấy frame ở vị trí lẻ). Biểu thức đảo ngược sẽ là $Rev_2(\sigma) = Z_2\left(T_2^1(\sigma), T_2^0(\sigma)\right) = \big((\sigma(1),\sigma(0)),(\sigma(3),\sigma(2)),(\sigma(5),\sigma(4)),(\sigma(7),\sigma(6),\dots)\big)$. Nghĩa là, Rev_2 đảo ngược thứ tự trong từng khối kích thước 2, mỗi cặp frame $(\sigma(0),\sigma(1))$ của stream σ trở thành $(\sigma(1),\sigma(0))$.

2.3.5. Đồng đại số stream dùng cho stream dạng BDL

Định nghĩa 2.14 (Máy trạng thái stream dạng BDL): Cho một tập hữu hạn hoặc vô hạn các phép toán xử lý stream dạng BDL, ký hiệu T. Một stream dạng BDL trong IoMT là một máy trạng thái stream có kiểu $\langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle$ bao gồm một tập các phép toán xử lý stream T, một tập các trạng thái *IoMT*, một hàm đầu ra o_{IoMT}: $IoMT \to (T \to 2)$, và một hàm tiến hóa e_{IoMT}: $IoMT \to (T \to IoMT)$.

Trong đó:

- $2 = \{0,1\}$ là một tập gồm 0 (ám chỉ phép toán t không khả dụng) hoặc 1 (ám chỉ phép toán t khả dụng).

- o_{IoMT} gán cho một trạng thái c, một hàm đầu ra $o_{IoMT}(c): T \to 2$ đặc tả giá trị $o_{IoMT}(c)(t)$ đạt được sau khi một phép toán t được thực hiện. Nói một cách khác,

$$o_{IoMT}(c)(t) = \begin{cases} 1\text{: nghĩa là khi phép toán } t \text{ khả dụng} \\ 0\text{: ngược lại} \end{cases}$$

- Tương tự, e_{IoMT} gán cho mỗi trạng thái c, một hàm tiến hóa $e_{IoMT}(c): T \to IoMT$ đặc tả trạng thái $e_{IoMT}(c)(t)$ đạt được sau khi một phép toán t được thực hiện. Đôi khi, $c \xrightarrow{t} c'$ được sử dụng để chỉ $e_{IoMT}(c)(t) = c'$.

Dựa vào đồng đại số stream ở tiểu mục 1.6.9 (Chương 1) và định nghĩa 2.14 ở trên, luận án xây dựng định nghĩa 2.15 về đồng đại số stream dùng cho BDL sau:

Định nghĩa 2.15 (Đồng đại số stream dạng BDL): Tập các stream $\{\langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle\}$ dạng BDL của tất cả các vật mang stream hữu hạn hoặc vô hạn có một cấu trúc đồng đại số gồm có một cặp phép toán $\langle O, E\rangle$ có dạng $\{\langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle\} \xrightarrow[\langle O,E\rangle]{} o_{IoMT} \times \{\langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle\}$ và phần tử stream $\langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle \longmapsto \langle\langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle(0), \langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle'\rangle$ (Bảng 2.21).

Trong đó:

- $\{\langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle\}$ là tập các stream, và o_{IoMT} là tập các frame.
- $\{\langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle\} \xrightarrow[\langle O\rangle]{} o_{IoMT}$ là cấu xạ của stream $\langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle$ đến cặp $\langle\langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle(0), \langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle'\rangle$, với $\langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle(0)$ là phần tử đầu tiên của stream $\langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle$ và $\langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle'$ là stream con của $\langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle$ bắt đầu từ phần tử thứ hai.
- $\{\langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle\} \xrightarrow[\langle E\rangle]{} \{\langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle\}$ là một cấu xạ của stream $\langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle$ đến stream $\langle E(\langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle)\rangle$, với $\langle E\big(\langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle(0)\big)\rangle$ được xác định bởi $\langle E\big(\langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle(0)\big) = \langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle(0)\rangle$ và $\langle E(\langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle)' = \langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle'\rangle$. Tiếp tục, $\langle E(\langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle)\rangle \xrightarrow[\langle E\rangle]{} \langle E(\langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle)\rangle$ là cấu xạ của một stream $\langle E(\langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle)\rangle$ đến stream $\langle E(\langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle)\rangle$, với $\langle E\big(\langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle(1)\big)\rangle$ được xác định bởi $\langle E\big(\langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle(1)\big) = \langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle(1)\rangle$ và $\langle E(\langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle)'' = \langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle''\rangle$.

Bảng 2.21. Một cấu trúc đồng đại số stream dùng cho BDL trong IoMT

Cặp phép toán $\langle O, E\rangle$	Phần tử stream dạng BDL
$\{\langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle\} \xrightarrow[\langle O,E\rangle]{} o_{IoMT} \times \{\langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle\}$	$\langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle \longmapsto \langle\langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle(0), \langle IoMT, \langle o_{IoMT}, e_{IoMT}\rangle\rangle'\rangle$

Ví dụ 2.14: Cho một stream dạng BDL trong IoMT: $\sigma = (\sigma(0), \sigma(1), \sigma(2), \sigma(3), \sigma(4), \sigma(5), \sigma(6), \sigma(7), \ldots)$ và thực hiện phân tích frame *đầu (head)* và các frame *đuôi (tail)* của stream σ này bằng cấu trúc đồng đại số stream như sau:

Đầu:	$\sigma(0)$	*Đuôi:*	$\sigma(1), \sigma(2), \sigma(3), \sigma(4), \sigma(5), \sigma(6), \sigma(7), \ldots$
	$\sigma(1)$		$\sigma(2), \sigma(3), \sigma(4), \sigma(5), \sigma(6), \sigma(7), \ldots$
	$\sigma(2)$		$\sigma(3), \sigma(4), \sigma(5), \sigma(6), \sigma(7), \ldots$
	$\sigma(3)$		$\sigma(4), \sigma(5), \sigma(6), \sigma(7), \ldots$

$\sigma(4)$	$\sigma(5), \sigma(6), \sigma(7), \ldots$
$\sigma(5)$	$\sigma(6), \sigma(7), \ldots$
$\sigma(6)$	$\sigma(7), \ldots$

2.3.6. Bài toán mô phỏng và kiểm chứng đánh giá các phép toán xử lý stream

Bài toán kết hợp các phép toán taking, zipping, reversing, dropping, splitting và merging dùng cho stream dạng BDL (Một stream có nhiều frame và mỗi frame là một bộ ⟨text, image, audio, video, v.v⟩, theo định nghĩa 2.13) nhằm: (1) Kiểm chứng ngữ nghĩa để bảo toàn tính đồng nhất của stream gốc và giữ tập con của phép toán stream này; (2) Lượng hóa các chỉ số hiệu năng chính (Key performance indicators - KPIs) của hệ thống khi sử dụng 6 phép toán: *Thông lượng (Throughput - TP), độ trễ trung bình (Latency average - LA), độ trễ bách phân vị (Latency p95 - LP), mức chiếm dụng bộ đệm trung bình (Buffer average - BA), đỉnh bộ đệm (Buffer peak - BP), chỉ số đo cân bằng (Fairness Index- FI) khi có nhiều stream đồng thời*, và *tỉ lệ tiết kiệm nhập/xuất (Input/Output - IO)*.

a) Đặt bài toán cho các phép toán cần kiểm chứng

- Phép toán taking $T_l^i: \mathbb{A}^\omega \to \mathbb{A}^\omega$ lấy mỗi frame cách nhau l khối, bắt đầu từ vị trí i:
$$T_l^i(\sigma)(n) = \sigma(i + nl).$$
- Phép toán zipping $Z_k: (\mathbb{A}^\omega)^k \to \mathbb{A}^\omega$ là ghép k stream bằng cách thực hiện xen kẽ vòng tròn (Round-robin):
$$Z_k(\sigma_0, \ldots, \sigma_{k-1})(n) = \sigma_r(q), n = qk + r\ (0 \le r < k).$$
- Phép toán reversing $Rev_k: \mathbb{A}^\omega \to \mathbb{A}^\omega$, với $k \ge 1$ được tổ hợp từ phép toán zipping và taking thành biểu thức stream như sau: $Rev_k(\sigma) = Z_k\left(T_k^{k-1}(\sigma), T_k^{k-2}(\sigma), \cdots, T_k^0(\sigma)\right)$.
- Phép toán dropping $D_l^k: \mathbb{A}^\omega \to \mathbb{A}^\omega$ mang nhiệm vụ loại bỏ phần tử thứ k của từng khối stream đầu vào với kích thước khối l, $l \ge 2$ và $0 \le k < l$.
- Phép toán splitting $S(\sigma)^{(k)} = S\left(\sigma^{(l)}\right)$, $S(\sigma)(j) = \sigma\left(n_j\right)$, $0 \le j < k$, $l > 1$ và $1 \le k \le l$, $S(\sigma) = Z_k\left(T_l^{n_0}(\sigma), T_l^{n_1}(\sigma), \cdots, T_l^{n_{k-1}}(\sigma)\right)$.
- Phép toán merging $\sigma + \tau = (\sigma(0) + \tau(0), \sigma(1) + \tau(1), \sigma(2) + \tau(2), \ldots)$, $\sigma = (\sigma(0), \sigma(1), \sigma(2), \ldots)$ và $\tau = (\tau(0), \tau(1), \tau(2), \ldots)$.
- Vậy, hai tính chất sau đây cần kiểm chứng:
  - *Đồng nhất (Identity):* Tách một stream σ thành l nhánh (khối) con như sau $T_l^0(\sigma), \ldots, T_l^{l-1}(\sigma)$, rồi zipping lại sẽ khôi phục đúng stream gốc:
$$Z_l\left(T_l^0(\sigma), T_l^1(\sigma), \ldots, T_l^{l-1}(\sigma)\right) = \sigma.$$
  - *Giữ một tập con (Subset retention):* Nếu chỉ chọn tập con chỉ số $I = \{i_0, i_1, \ldots, i_{k-1}\}$, rồi zipping các khối l tương ứng thu được stream đầu ra có tần số giảm còn $\rho = \frac{i}{l}$ so với stream gốc ($\rho = 1$ giữ lại toàn bộ frame). Nghĩa là mỗi khối l chỉ giữ lại i frame, giúp giảm thông lượng, tiết kiệm IO và ngăn dồn ứ frame khi hệ thống quá tải.
- Phép toán take, zip, rev, drop, split và merge tác động lên các KPIs.

b) Phân tích đầu vào và đầu ra cho thuật toán mô phỏng các phép toán stream

- **Đầu vào:** Của mô phỏng không phải là tập tin bên ngoài mà là các tham số cấu hình (Có giá trị tùy ý) và frame sinh ra từ mô hình.
  - **Tham số cấu hình:** Người dùng cung cấp tham số cấu hình (Các tham số này xác định

cách frame được sinh ra và được lấy bởi phép toán taking và zipping ở lối vào pipeline; Những frame được chọn sẽ đi qua pipeline: Decode giải mã dữ liệu thô (video, audio, image, text) → Process thực hiện xử lý tính toán trên frame đã giải mã → Encode mã hoá kết quả để xuất ra stream đầu ra).

- T: Tổng thời gian mô phỏng (Đơn vị tính theo giây - s).
- Dt: Bước thời gian rời rạc (Đơn vị tính theo mili giây - ms).
- M: Số lượng nguồn stream đầu vào: 0..M-1.
- $\mathrm{fps_{in}}$: Tốc độ frame đầu vào mỗi giây cho từng nguồn stream. Khoảng cách giữa hai frame liên tiếp của nguồn i là $\frac{1}{\mathrm{fps_{in}}[i]}$. Giả sử $\mathrm{fps_{in}}[i] = \{60, 45, 30\}$ là 3 nguồn.
- l: Kích thước nhóm (khối) cho phép toán Taking. Mỗi frame có chỉ số Idx, xét $\mathrm{Idx} \bmod l \in \{0, \dots, \mathrm{k}-1\}$ để quyết định giữ hay loại frame theo I.
- I: Danh sách các chỉ số trong nhóm (khối) l được giữ lại, giả sử có danh sách các chỉ số gồm [1, 3] (Nghĩa là chỉ giữ lại frame 1 và 3).
- Policy: Chính sách khi nguồn rỗng (Xử lý trong bước nén đa nguồn khi không có frame sẵn): Blocking: Dừng nếu nguồn hiện tại chưa có frame; Skip: Bỏ qua nguồn rỗng, lấy từ nguồn kế tiếp; Padding: Nếu tới lượt mà nguồn trống thì chèn frame đệm hay còn gọi là frame ảo để giữ nhịp phát).

- **Dữ liệu frame được sinh ra từ mô hình:** Trong quá trình mô phỏng, các frame được tạo ra cho từng nguồn (Đây chính là các frame đầu vào được mô phỏng liên tục trong thời gian chạy): Mỗi frame có cấu trúc:
  - Source: Chỉ số nguồn stream σ_M (0, 1, 2, …, M-1), $M \geq 1$.
  - Idx: Số thứ tự frame của nguồn stream.
  - Tarrive: Thời điểm frame đến hệ thống.
  - Padding: Bật, nếu là frame ảo (frame đệm) thì không đưa vào decode.

▪ **Đầu ra:** Khi chạy mô phỏng sinh ra lớp dữ liệu các chỉ số hiệu năng chính (KPIs) được dùng để tính toán (lượng hóa) sau khi kết thúc mô phỏng.

- *Thông lượng đầu ra (TP):* Cho biết hệ thống thực tế xuất ra bao nhiêu frame mỗi giây sau khi thực hiện phép toán taking và phép toán zipping.

$$\mathrm{TP} = \frac{\text{Số lượng frame đầu ra}}{\mathrm{T}}.$$

  - T: Tổng thời gian mô phỏng. Cho biết tốc độ xử lý trung bình.
- *Độ trễ trung bình (LM):* Của cơ chế đường ống (Pipeline), nghĩa là tính thời gian trung bình từ lúc frame đến cho đến khi frame được xuất ra (ms: mili giây). Giả sử có N frame với độ trễ từng frame là $L_1, L_2, \dots, L_N$, độ trễ của frame thứ i được tính $L_i = \mathrm{t_{out}}(\mathrm{f}_i) - \mathrm{t_{in}}(\mathrm{f}_i)$, thì độ trễ trung bình là:

$$\mathrm{LM} = \frac{1}{N}\sum_{i=1}^{N} L_i = \frac{1}{N}\sum_{i=1}^{N}\big(\mathrm{t_{out}}(\mathrm{f}_i) - \mathrm{t_{in}}(\mathrm{f}_i)\big).$$

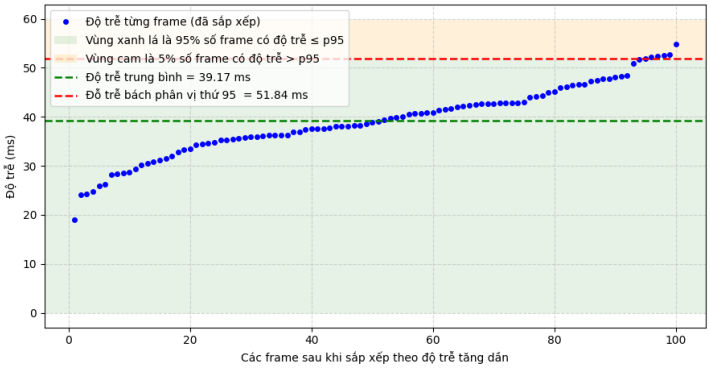
  - Với mỗi frame f_i, tính từ lúc frame vào đến lúc ra, $\mathrm{t_{in}}(\mathrm{f}_i)$ là thời điểm frame vào pipeline (là Tarrive), $\mathrm{t_{out}}(\mathrm{f}_i)$ là thời điểm frame ra khỏi Encode. Độ trễ bao gồm thời gian chờ trong hàng đợi cộng với thời gian xử lý tại mỗi giai đoạn pipeline.
  - Giả sử có 100 giá trị độ trễ $L_1, L_2, \dots, L_{100}$. Công thức $\mathrm{LM} = \frac{1}{100}\sum_{i=1}^{100} L_i$. Giả sử các giá trị độ trễ đã đo lần lượt là $L_i = [20.1, 21.4, 23.5, \dots, 58.2, 59.7]$, giả sử frame đầu tiên f_1 áp

dụng $L_1 = t_{out}(f_1) - t_{in}(f_1) = 120.1 - 100.0 = 20.1$ ms (Độ trễ của frame đầu tiên được tính bằng chênh lệch giữa thời điểm frame đó ra khỏi và vào pipeline). Tổng tất cả độ trễ $\sum_{i=1}^{100} L_i = 3917$. Khi đó LM $= \frac{3917.0}{100} = 39.17$ ms (LM của 100 frame) (Hình 2.7).

- *Độ trễ bách phân vị (LP):* Nghĩa là ngưỡng độ trễ, đây là giá trị mà 95% các frame (p95) có độ trễ ≤ giá trị này, và 5% số frame còn lại có độ trễ lớn hơn. Có N độ trễ $L_1, L_2, \dots, L_N$ đã sắp xếp tăng dần $L_1 \le L_2 \le \cdots \le L_N$ và phần trăm cần tính $p = 0.95$, khi đó độ trễ bách phân vị thứ 95 nội suy tuyến tính sẽ được tính là: $r = 1 + p \times (N-1), k = \lfloor r \rfloor, \gamma = r - k, LP = \begin{cases} L_r, \text{ nếu } \gamma = 0; \\ (1-\gamma) \times L_k + \gamma \times L_{k+1}, \text{nếu } 0 < \gamma < 1. \end{cases}$

  Trong đó:

  - Nếu $\gamma = 0$ (Tức r là số nguyên), $LP = L_r$, nghĩa là giá trị bách phân vị trùng đúng với một phần tử trong dãy. Ngược lại, nếu $0 < \gamma < 1$, thì LP được nội suy tuyến tính giữa hai giá trị liền kề L_k và L_{k+1}, theo $LP = (1-\gamma)L_k + \gamma L_{k+1}$.
  - N: Số lượng frame được đo, nghĩa là tổng số giá trị độ trễ thu được trong toàn bộ mô phỏng. Giả sử $N = 100$ nghĩa là có 100 frame được ghi nhận độ trễ.
  - p: Phần trăm vị cần tính, nghĩa là tỷ lệ phần trăm của bách phân vị. Với $p = 0.95$, ta tính độ trễ bách phân vị thứ 95, tức 95% frame có độ trễ ≤ giá trị này.
  - r: Vị trí thực trong dãy đã sắp xếp, nghĩa là tính theo công thức $r = 1 + p \times (N-1)$. Đây là vị trí thực của giá trị bách phân vị trong dãy độ trễ đã được sắp xếp tăng dần.
  - k: Phần nguyên của r, nghĩa là chỉ số nguyên gần nhất về phía dưới của r, tức $k = \lfloor r \rfloor$.
  - γ: Phần thập phân của r, nghĩa là phần dư giữa r và k, tức $\gamma = r - k$, cho biết khoảng cách tương đối giữa hai frame cần nội suy.
  - L_k: Độ trễ tại vị trí thứ k trong dãy đã sắp xếp, nghĩa là dãy độ trễ sau khi sắp xếp là $L_1 \le L_2 \le \cdots \le L_N$, khi đó L_k là giá trị tại vị trí thứ k trong dãy này.
  - L_{k+1}: Độ trễ tại vị trí ngay sau k trong dãy độ trễ đã được sắp xếp tăng dần, nghĩa là khi ta có dãy độ trễ đã sắp xếp: $L_1 \le L_2 \le \cdots \le L_N$, thì L_{k+1} là giá trị đứng ngay sau L_k, tức phần tử kế tiếp trong dãy.
  - Minh họa $N = 100$ frame và $p = 0.95 \rightarrow r = 95.05 \rightarrow k = 95, \gamma = 0.05, L_{95} = 51.8$ ms, $L_{96} = 52.6$ ms. Khi đó: LP $= (1 - 0.05) \times 51.8 + (0.05 \times 52.6) = 51.84$ ms (Hình 2.7).



Hình 2.7. So sánh giữa LM và LP của độ trễ (100 frames)

  - Thực hiện so sánh giữa *LM* và *LP* của độ trễ bằng kết quả đồ thị trực quan (Hình 2.7): Hai đường gạch đứt nét biểu diễn *LM* và *LP* trong pipeline. *LM* phản ánh độ trễ trung bình của toàn bộ pipeline, còn *LP* thể hiện ngưỡng mà 95% số frame được xử lý nhanh hơn, chỉ 5%

còn lại nằm ở phần đuôi phân bố, tức nhóm frame bị trễ nhất trong chuỗi xử lý. Vậy, chỉ số p95 được dùng để đánh giá độ ổn định và khả năng chịu tải của pipeline.

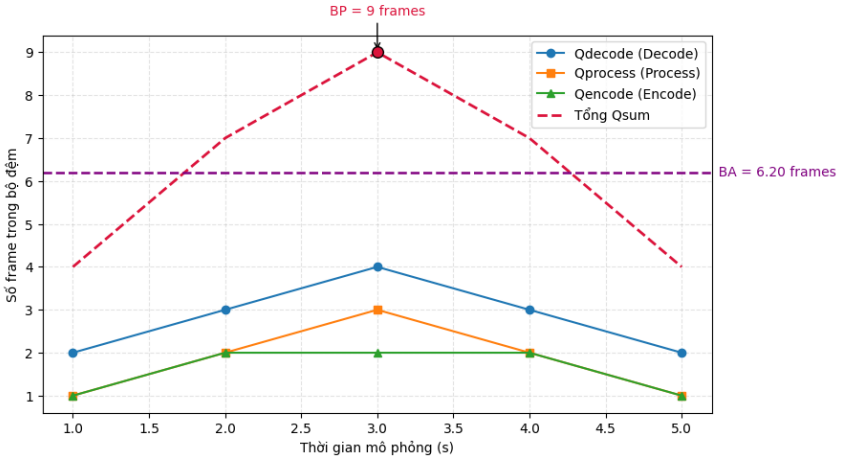
- *Mức chiếm dụng bộ đệm trung bình (BA):* Nghĩa là giá trị trung bình cộng của số frame đang chờ trong các bộ đệm trong suốt thời gian mô phỏng cho thấy hệ thống có bị dồn ứ frame hay không. Tổng dung lượng bộ đệm trung bình (Cộng 3 giai đoạn của pipeline Decode → Process → Encode).

$$\mathrm{BA} = \frac{1}{\mathrm{T}} \sum_0^{\mathrm{T}} \mathrm{Dt}\left(\mathrm{Q_{decode}} + \mathrm{Q_{process}} + \mathrm{Q_{encode}}\right).$$

  - Với $Q_{\text{giai đoạn}}$ là số frame trong bộ đệm tại thời điểm t. Bộ đệm là trạng thái hàng đợi theo mỗi bước thời gian Dt, mỗi $\mathrm{Q_{giai\ đoạn}}$ là frame đang xếp hàng tại bước thời gian t.
  - Giả sử thời gian mô phỏng là $T = 5\ s$ (Tức 5 chu kỳ quan sát) và mỗi chu kỳ có giá trị số frame đang chờ trong các bộ đệm ở bảng 2.22.

Bảng 2.22. Số lượng frame trong các bộ đệm ở từng chu kỳ thời gian ($T = 5\ s$)

Thời điểm (s)	$\mathrm{Q_{decode}}$	$\mathrm{Q_{process}}$	$\mathrm{Q_{encode}}$	Tổng $\mathrm{Q_{sum}}$
1	2	1	1	4
2	3	2	2	7
3	4	3	2	9
4	3	2	2	7
5	2	1	1	4



Hình 2.8. Biểu đồ chi tiết mức chiếm dụng bộ đệm của từng giai đoạn

  - Giả sử $Dt = 1\ s$, $\mathrm{BA} = \frac{1}{\mathrm{T}} \sum_0^{\mathrm{T}} \mathrm{Dt}\left(\mathrm{Q_{decode}} + \mathrm{Q_{process}} + \mathrm{Q_{encode}}\right) = \frac{1}{5}(4 + 7 + 9 + 7 + 4) = 6.2$. Vậy, trung bình trong suốt thời gian mô phỏng, hệ thống có 6.2 frame đang chờ trong pipeline (Hình 2.8). Mức này thể hiện độ tải trung bình của hệ thống, càng thấp thì pipeline càng thông suốt và ít dồn ứ. Nghĩa là đo độ tải trung bình của hệ thống.

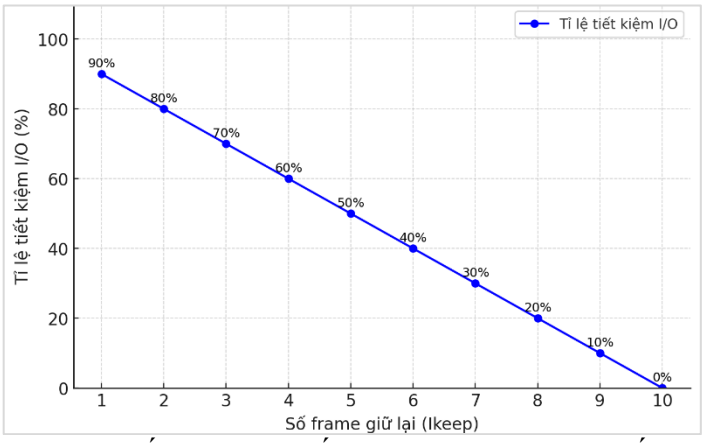
- *Đỉnh bộ đệm (BP):* Nghĩa là số frame tối đa từng dồn ứ trong bộ đệm (Đỉnh bộ đệm cao nhất từng đạt). Nếu giá trị này cao, thì hệ thống dễ gặp dồn ứ frame và độ trễ tăng mạnh: $\mathrm{BP} = \max_{\mathrm{t}}\left(\mathrm{Q_{decode}} + \mathrm{Q_{process}} + \mathrm{Q_{encode}}\right)$.
  - Với $Q_{\text{Giai đoạn}}$ là số frame trong bộ đệm tại thời điểm t. Dựa vào bảng 2.22, luận án có tổng số frame trong bộ đệm tại mỗi thời điểm t là: $\mathrm{Q_{sum}} = [4, 7, 9, 7, 4]$. Áp dụng công thức tính $\mathrm{BP} = \max_{\mathrm{t}}\left(\mathrm{Q_{decode}} + \mathrm{Q_{process}} + \mathrm{Q_{encode}}\right) = max(4, 7, 9, 7, 4) = 9$. Vậy, đỉnh bộ đệm $\mathrm{BP} = 9$ frames, nghĩa là tại thời điểm cao nhất, có 9 frame đồng thời bị xếp hàng trong pipeline (Hình 2.8). Nếu giá trị này cao, có nguy cơ nghẽn pipeline và độ trễ tăng mạnh. Ý

nghĩa là đo ngưỡng dồn ứ cực đại (điểm tắc nghẽn).

- *Chỉ số đo cân bằng (Fairness Index - FI):* Nghĩa là thước đo độ cân bằng trong phân phối tài nguyên (1: Hoàn toàn cân bằng).

$$\text{FI} = \frac{\left(\sum_{i=1}^{M} x_i\right)^2}{M \times \sum_{i=1}^{M} x_i^2}, \text{ với } M \geq 1.$$

  - x_i: Số frame thực tế được trả ra của nguồn i. Thang điểm 0–1, 1 là cân bằng.
    - Cho $M = 1, x_1 = 60$. Tính $\text{FI} = \frac{\left(\sum_{i=1}^{M} x_i\right)^2}{M \times \sum_{i=1}^{M} x_i^2} = \frac{(x_1)^2}{1 \times x_1^2} = \frac{(60)^2}{1.60^2} = 1$. Do vậy, khi chỉ có một nguồn (M = 1), hệ thống không cần so sánh mức phân phối giữa các nguồn, nên chỉ số cân bằng FI luôn bằng 1, thể hiện hoàn toàn cân bằng.
    - Cho $M = 2, x_i = \{60,45\}$. Tính $\text{FI} = \frac{\left(\sum_{i=1}^{M} x_i\right)^2}{M \times \sum_{i=1}^{M} x_i^2} = \frac{(105)^2}{2 \times (60^2 + 45^2)} = 0.98$. Do vậy, chỉ mức độ cân bằng giữa hai nguồn đạt 98%, tức là khá cân bằng (Nguồn 1 nhiều hơn một chút so với nguồn 2).
    - Cho $M = 3, x_i = \{60,45,30\}$. Tính $\text{FI} = \frac{\left(\sum_{i=1}^{M} x_i\right)^2}{M \times \sum_{i=1}^{M} x_i^2} = \frac{(135)^2}{3 \times (60^2 + 45^2 + 30^2)} = 0.931$. Do vậy, chỉ số cân bằng giữa 3 nguồn là 93.1%, nghĩa là mức độ công bằng khá cao, nhưng vẫn có sự chênh lệch do số frame lần lượt là 60–45–30.



Hình 2.9. Mối quan hệ tuyến tính giữa số frame giữ lại và mức tiết kiệm IO ($l = 10$).

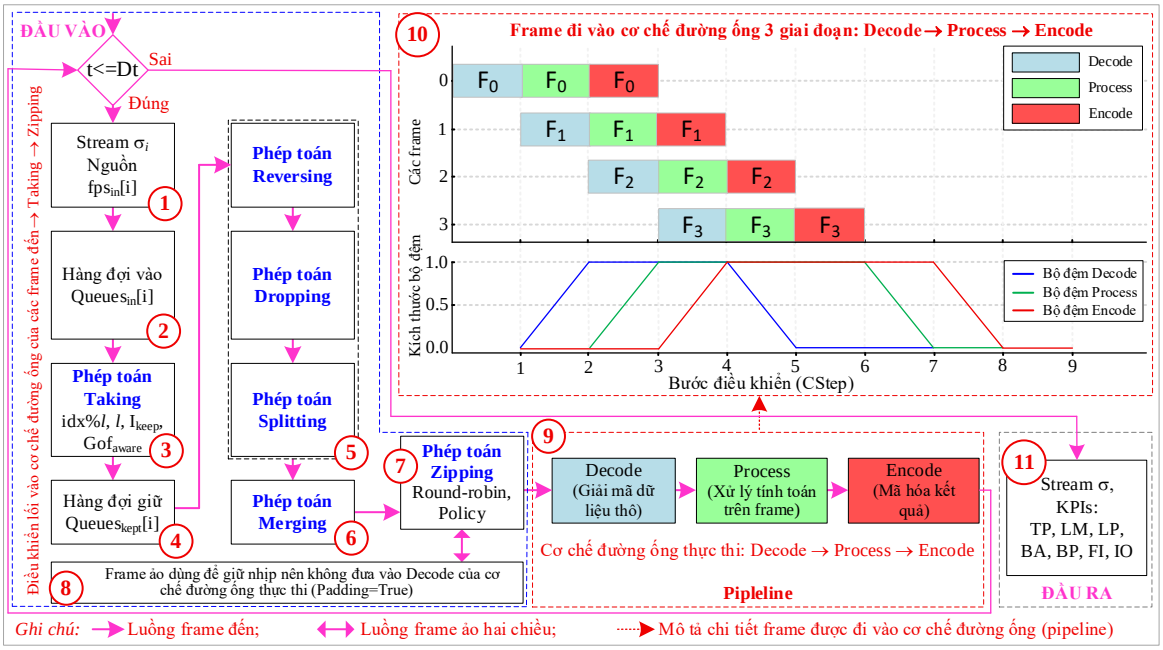
- *Tỉ lệ tiết kiệm nhập/xuất (IO):* So với truyền toàn bộ frame được tính theo công thức cho phép toán taking (Hình 2.9).

$$\text{IO} = 1 - \left(\frac{|I_{keep}|}{l}\right).$$

  - l: Kích thước khối (Số frame trong một khối) của phép toán taking; $|I_{keep}|$: Số frame được giữ lại trong khối l của phép toán taking. Khi giữ ít frame thì tỉ lệ tiết kiệm I/O cao. Khi giữ nhiều frame thì tỉ lệ tiết kiệm I/O giảm dần (Hình 2.9).

Hình 2.10, mỗi bước thời gian Dt, Khối ① có stream σ_i các nguồn sinh frame với tốc độ $\text{fps}_{\text{in}}[i]$ (Giả sử có $\text{fps}_{\text{in}}[i]$ = [60, 45, 30] nghĩa là ba nguồn lần lượt phát 60/45/30 mỗi giây). Vòng lặp xử lý theo trật tự các frame đến (Arrivals) → Khối ② đưa vào hàng đợi $\text{Queues}_{\text{in}}[i]$ → Khối ③ Taking → Khối ④ $\text{Queues}_{\text{kept}}[i]$ → Khối ⑤ (Reversing, Dropping, Splitting) → Khối ⑥ Merging → Khối ⑦ Zipping → Khối ⑨ pipeline (Decode → Process → Encode), sau mỗi vòng lặp cập nhật bộ đệm và tăng bước thời gian t += Dt. Phép toán taking và zipping là lớp điều tiết lối vào pipeline

của các frame đến, thực thi thật sự chỉ bắt đầu từ Decode.



Hình 2.10. Sơ đồ điều khiển lối vào cơ chế đường ống (Pipeline) của các frame đến

Khối ③ có phép toán taking không phải một giai đoạn pipeline mà là bộ lấy frame theo từng nguồn: Với mỗi frame đến, tính $j = idx \bmod l$ và giữ khi $j \in I_{keep}$, nếu bật nhận biết Gof_{aware} thì mọi frame nằm trong Avoidtypes (Thường là I_{frame}, I_{frame} là frame nội mã hoá làm mốc tham chiếu, không phụ thuộc frame trước hoặc frame sau, mốc tham chiếu trong nhóm các frame Gof - Group of frames) luôn được giữ. Các frame bị loại được ghi vào nhật ký các frame bị loại Drops, còn frame được chấp nhận đưa vào hàng đợi giữ trong Khối ④ $Queues_{kept}[i]$ theo nguồn. Phép toán taking không có hàng đợi hay thời gian phục vụ riêng, nó chỉ quyết định giữ hay bỏ frame, nhờ đó định hình lưu lượng và thành phần nội dung đi vào pipeline và đồng thời giảm IO. Khối ⑤ nhận nguồn từ khối ④ thực hiện đảo ngược, loại bỏ và tách frame, Khối ⑥ thực hiện trộn frame.

Khối ⑦ có phép toán zipping là cơ chế nén frame đa nguồn đặt trước pipeline thực thi. Ở mỗi bước thời gian Dt, nó duyệt các nguồn theo xen kẽ vòng tròn (Round-robin) và chọn tối đa một frame thật từ Khối ④ $Queues_{kept}[i]$ để đưa vào Decode. Vì thế, zipping quyết định frame của nguồn nào được đi trước và chính xác lúc nào frame đó được bơm vào pipeline. Cách chọn này còn phụ thuộc vào chính sách (Policy): với *Skip* nếu đến lượt mà nguồn đang rỗng thì bỏ qua ngay đi sang nguồn kế tiếp nên frame của nguồn khác có thể vào sớm hơn; với *Blocking* nếu nguồn rỗng thì cả bước đó không frame được chọn khiến thời điểm vào bị trì hoãn; Khối ⑧ với *Padding* khi nguồn rỗng sẽ tạo một frame ảo chỉ để giữ nhịp vòng lặp và frame ảo này không đi vào Decode. Vì quyết định thứ tự và thời điểm cấp frame, nên zipping tác động trực tiếp đến tải hàng đợi, độ trễ, cân bằng giữa nguồn và gián tiếp đến thông lượng.

Khối ⑩ trong từng bước thời gian, các frame F_0–F_3 đi vào pipeline ba giai đoạn Khối ⑨ Decode → Process → Encode, với trục ngang là các bước điều khiển tuần tự (CStep) và trục dọc là các frame; F_0 (Chạy tuần tự qua 3 giai đoạn) chuyển sang Process thì F_1 bắt đầu Decode, tiếp đó F_2, F_3 nối tiếp, tạo xử lý chồng lấn giúp giảm độ trễ tổng và tăng tốc xử lý. Khi nhiều frame đến cùng lúc, bộ đệm Decode phình ra trước khi Process kịp tiêu thụ; lúc Process bận, hàng đợi chờ của nó dồn; Encode cũng có thể dồn nếu tốc độ xuất chậm hơn tốc độ vào. Nói tổng quát, mỗi frame đều đi qua ba giai đoạn Khối ⑨ nhưng không phải chờ nhau hoàn tất; Taking quyết định *cái gì* được vào (Giữ lại, giảm I/O), Zipping quyết định khi nào và theo thứ tự nào từng nguồn được cấp vào Decode (Chi phối nhịp phát và cân bằng), còn pipeline quyết định được xử lý như thế nào với thời gian phục vụ và bộ đệm của từng giai đoạn, ba lớp (Khối ③, Khối ⑦, Khối ⑨) phối hợp để định hình chi phối Khối ⑪ TP, LM, LP, BA, BP, FI và IO. Đây chính là ưu điểm then chốt của lập lịch theo pipeline cho stream dạng BDL, tận dụng song song hóa để giảm độ trễ tổng và tăng tốc độ xử lý [CT.9].

c) Thuật toán mô phỏng các phép toán xử lý stream

Thuật toán mã giả sau đây mô tả logic mô phỏng kết hợp các phép toán xử lý stream cho việc xử lý điều khiển lối vào pipeline của các frame đến.

Thuật toán 2.1: Điều khiển lối vào cơ chế đường ống (pipeline) cho các frame đến bằng các phép toán xử lý stream (Một stream hoặc nhiều stream) qua 3 giai đoạn.

Đầu vào:

- M stream vào hệ thống $\{\sigma_0, \sigma_1, \dots, \sigma_{M-1}\}$, mỗi frame là một bộ ⟨text, image, audio, video, v.v⟩
- T là tổng thời gian mô phỏng (Giây), Dt là bước thời gian (Mili giây)
- $\text{fps}_{\text{in}}[i]$ là tốc độ frame đầu vào mỗi giây cho từng nguồn stream σ_i
- l là kích thước khối, $I \subseteq \{0,1,\dots,k-1\}$ là các vị trí được giữ trong phép toán stream
- $\text{Gof}_{\text{aware}}$ là bật/tắt chế độ nhận biết frame khi Taking. Nếu bật: Nếu một frame thuộc loại trong Avoidtypes (Các loại frame cần bảo vệ khi $\text{Gof}_{\text{aware}}$=True, mặc định là I, còn P cho các vị trí còn lại, suy ra bảo vệ I_{frame}) và lẽ ra bị loại theo I_{keep}, nó vẫn được giữ để bảo toàn (Bảo vệ I_{frame}); Ngược lại nếu tắt thì theo luật I_{keep}.
- Gof_{size}: Kích thước nhóm frame (GoF).
- Chính sách nén (Zipping) khi đến lượt một nguồn thuộc 3 loại Policy {Blocking, Skip, Padding}:
  - Blocking: Nếu nguồn đang đến lượt không có frame, không chọn nguồn khác trong bước thời gian đó (Giữ nhịp chặt, thông lượng có thể giảm).
  - Skip: Nếu nguồn đang đến lượt trống, thử nguồn kế tiếp ngay (Thông lượng thường cao hơn, có thể giảm cân bằng).
  - Padding: Nếu nguồn đang đến lượt trống, tạo một frame đệm/ảo (Padding=True) để giữ nhịp đồng bộ (nhịp phát) và rồi sau đó chuyển lượt.

Đầu ra: Stream σ_i, KPIs (TP, LM, LP, BA, BP, FI, IO)

1. Khởi tạo ngẫu nhiên dữ liệu nếu cần, đặt thời gian $t \leftarrow 0$
2. for each nguồn i:
3. Tạo hàng đợi vào $\text{Queue}_{\text{in}}[i]$
4. Đặt $\text{Arrival}_{\text{next}}[i] \leftarrow 0$ là bước khởi tạo thời điểm đến kế tiếp của nguồn i (Ở thời điểm bắt đầu mô phỏng, t = 0 mili giây, nguồn i đã sẵn sàng sinh ra frame đầu tiên).
5. Tạo hàng đợi cho các giai đoạn theo pipeline: Q_{Decode}, Q_{Process}, Q_{Encode}
6. while $t < T$:

```
for each i in 0..M-1: Sinh ra danh sách frame đến từ M stream vào hệ thống {σ_0, σ_1, ..., σ_{M-1}}
  if t ≥ Arrival_next[i]:
    Frame ← Frame mới được tạo (source=i, t_arrive(t_in)=t, idx=counter[i], Gof_pos=counter[i]
            mod Gof_size, Ftype=FrameType(Gof_pos))
    Queue_in[i] ← Frame: Frame xếp hàng (Enqueue) đó để vào Queue_in[i]
    Arrival_next[i] ← Arrival_next[i] + 1/fps_in[i]: Thời điểm frame đến hệ thống
    Counter[i] ← Counter[i] + 1
for each i: Áp dụng phép toán Taking cho từng nguồn
  while Queue_in[i] không rỗng:
    f ← Queue_in[i].pop(0)
    j ← f.idx mod l
    Keep ← (j ∈ I_keep)
    if Gof_aware và (f.Ftype ∈ Avoidtypes) và không keep:
        Keep ← True là bảo vệ I_frame khi Gof_aware = True
    if keep:
        Queue_kept[i].append(f)
    else:
        Drop f: Bỏ (không giữ) frame f vì nó không thỏa điều kiện giữ (j ∉ I_keep). Frame này sẽ không đi tiếp
        vào pipeline; Ghi lại một dòng nhật ký về sự kiện bị bỏ này (Thời điểm t, nguồn source i, chỉ mục idx,
        loại ftype), để sau này tính KPIs (như tiết kiệm IO, phân tích lý do rơi frame, v.v.).
  Dequeue Queue_in[i]: Lấy f ra khỏi hàng đợi vào của nguồn i (Dù giữ hay bỏ thì đều phải loại khỏi Queue_in[i]
  để tiến tới frame kế tiếp).
if use_reversing = true or use_dropping = true or use_splitting = true then
  for i from 0 to M - 1 do
    if Queue_kept[i] rỗng then                  //Nếu stream i rỗng thì bỏ qua
      continue
    endif
    block ← Queue_kept[i]                       //Block là danh sách frame của stream i
    if use_reversing = true then                //Reversing operator Rev_k
      k ← rev_k
      if k > 1 then
        out ← Danh sách rỗng
        n ← Độ dài (block)
        for j from 0 to n - 1 step k do         //Chia block thành từng "khối con" liên tiếp độ dài k
          sub ← block[j:j + k]                  //Cắt segment từ j, tối đa k phần tử
          out ← out + Reverse(sub)              //Đảo ngược thứ tự trong sub rồi nối vào out
        endfor
        block ← out
      endif
    endif
    if use_dropping = true then                 //Dropping operator D_l^k
      l ← drop_l                                //Độ dài khối
      dk ← drop_k                               //Vị trí sẽ drop trong mỗi khối (0-based)
      if l > 1 then
        out ← Danh sách rỗng
        n ← Độ dài (block)
        for j from 0 to n - 1 step l do         //Chia block thành từng khối con độ dài l
          sub ← block[j:j + l]
          if dk < Độ dài (sub) then
            for h from 0 to độ dài(sub) - 1 do //Bỏ phần tử tại vị trí dk, giữ các phần tử còn lại
              f ← sub[h]
```

```
                        if h ≠ dk then
                            Append(out, f)
                        endif
                    endfor
                else
                    out ← out + sub          //Nếu dk ≥ |sub| thì không drop gì, giữ nguyên sub
                endif
            endfor
            block ← out
        endif
    endif
    if use_splitting = true và split_ns không rỗng then      //Splitting operator S
        l ← split_l                                          //Độ dài khối dùng cho Split
        n_list_sorted ← Sort(Unique(split_ns))
        if l > 1 then
            out ← Danh sách rỗng
            n ← Độ dài(block)
            for j from 0 to n - 1 step l do          //Chia block thành từng khối con độ dài l
                sub ← block[j:j + l]
                for mỗi pos trong n_list_sorted do   //Với mỗi vị trí pos trong n_list_sorted,
                    if pos < Độ dài (sub) then       //Nếu tồn tại trong sub thì lấy phần tử đó
                        Append(out, sub[pos])
                    endif
                endfor
            endfor
            block ← out
        endif
    endif
    Queue_kept[i] ← block            //Gán lại kết quả sau khi Rev/Drop/Split cho stream i
  endfor
endif
if use_merging = true and M > 1 then //Merging M stream frame gộp tất cả Queue_kept[0..M-1] thành 1 stream
  merged ← Danh sách rỗng
  turn_m ← 0                                  //Đánh dấu "lượt" của stream hiện tại
  still ← true                                //Cờ kiểm tra còn frame nào để merge hay không
  while still = true do
    still ← false
    for step from 1 to M do //Quét qua M stream, theo thứ tự xen kẽ vòng tròn, bắt đầu từ turn_m
        i ← turn_m
        if Queue_kept[i] không rỗng then
          f ← phần tử đầu tiên của Queue_kept[i] //Lấy frame đầu tiên trong hàng đợi i
          Xóa phần tử đầu tiên khỏi Queue_kept[i]
          turn_m ← (i + 1) mod M              //Chuyển lượt sang stream tiếp theo
        else
          turn_m ← (i + 1) mod M              //Hàng đợi i rỗng, chỉ xoay lượt sang stream tiếp theo
        endif
    endfor
  endwhile //Khi sau một vòng For mà tất cả Queue_kept[i] đều rỗng, biến still vẫn = false suy ra while dừng
  Queue_kept[0] ← merged            //Sau khi merge xong, chỉ giữ stream 0 là stream đã gộp
  for i from 1 to M - 1 do
    Queue_kept[i] ← []              //Danh sách rỗng
```

105. endfor
106. endif
107. chosen ← None *//Khởi tạo bắt đầu với trạng thái "chưa có frame được chọn"*
108. tries ← 0 *//Đếm số lần thử các nguồn*
109. while chosen = None và tries < M do *//Zipping operator(chọn mỗi frame mỗi bước)*
110. i ← (turn + tries) mod M
111. if $Queue_{kept}$[i] is not empty then
112. chosen ← pop_front($Queue_{kept}$[i]) *//Lấy frame thật*
113. turn ← (i + 1) mod M
114. else *//Chính sách xử lý khi nguồn trống*
115. if policy = "Blocking" then *//Chặn lại, không chọn frame nữa*
116. chosen ← None
117. break
118. else if policy = "skip" then *//Bỏ qua nguồn trống, thử nguồn kế tiếp*
119. tries ← tries + 1
120. else if policy = "Padding" then *//Tạo frame đệm để giữ nhịp*
121. chosen ← Frame(source = i, idx = -1, t_{arrive} = t, gof_pos = 0, ftype = 'P', padding = True)
122. turn ← (i + 1) mod M
123. end if
124. end if
125. end while
126. if chosen ≠ None và chosen.padding = False then *//Nếu chọn được frame thật suy ra đưa vào Decode*
127. Enqueue Q_{Decode} ← frame *//Với t_{zip}=t thời điểm được chọn nhập pipeline, rồi đưa frame đó vào Q_{Decode}).*
128. end if
129. Cơ chế đường: Decode → Process → Encode theo thời gian. Khi một frame hoàn tất ở giai đoạn Encode: Ghi nhận t_{out} và tính LM = t_{out} - t_{in}
130. Thống kê độ chiếm dụng bộ đệm của các hàng đợi Q_{Decode}, $Q_{Process}$, Q_{Encode}
131. t ← t + Dt *//Tăng bước thời gian*
132. Tính KPIs: TP, LM, LP, BA, BP, FI ($M \geq 1$), IO.
133. Xuất stream σ_i, KPIs

d) Vai trò của các phép toán xử lý stream trong kịch bản mô phỏng

Phép toán taking làm giảm tần số theo khối l và tập chỉ số I_{keep}. Trong vòng lặp thuật toán 2.1, phép toán taking lấy frame theo T_l^i ở tầng nguồn bằng cách cắt nhỏ stream theo khối l và chỉ giữ các frame trong I_{keep} theo pipeline thỏa *tính chất giữ một tập con (Tiểu mục 2.3.6 – mục a)*. Phép toán taking tác động lên các KPIs, gồm: (1) Tỉ lệ tiết kiệm IO đúng với mỗi khối l chỉ giữ lại I_{keep} frame giúp giảm tải vào pipeline; (2) Thông lượng đầu vào pipeline giảm xấp xỉ theo tỉ lệ $\frac{|I_{keep}|}{l}$ giúp giảm tắc nghẽn, kéo theo giảm BA, BP và LA, LP khi pipeline là nút cổ chai; (3) Đồng nhất khi giữ lại hết I_{keep} frame không làm biến đổi stream.

Phép toán zipping là ghép cân bằng nhiều stream *(Đã được lấy từ phép toán taking)* thành một stream duy nhất để đưa vào pipeline, kiểm soát nhịp vào pipeline, và xử lý thiếu frame bằng các chính sách: Blocking, Skip và Padding. Phép toán zipping tác động lên các KPIs, gồm: (1) Chỉ số đo cân bằng (FI) giữa nguồn từ phân bố số frame đầu ra của mỗi nguồn; (2) Xen kẽ vòng tròn tránh một nguồn lấy hết băng thông vào pipeline; (3) Điều tiết nhịp vào tối đa một frame trên mỗi mili giây giúp tách rời lũy kế frame đến với khả năng phục vụ pipeline.

Phép toán taking và zipping giải quyết chuỗi xử lý chọn lọc rồi xen kẽ nhằm khớp tải với năng lực pipeline, bảo toàn chất lượng quan trọng và giữ cân bằng giữa nhiều nguồn. Kết hợp taking và zipping, luận án có được biểu thức stream đầu vào pipeline như sau: $Z_M\left(T_l^i(\sigma_0), T_l^i(\sigma_1), \ldots, T_l^i(\sigma_{M-1})\right)$, trong đó T_l^i là giữ các vị trí trong I_{keep}, còn Z_M là xen kẽ trong giữa M nguồn thỏa *tính đồng nhất (Tiểu mục 2.3.6 – Mục a)*.

Trong pipeline BDL, các phép toán xử lý stream chịu trách nhiệm điều tiết và tái cấu trúc luồng frame trước khi vào các giai đoạn tính toán (Pipeline). Reversing đảo thứ tự frame theo chu kỳ k để mô phỏng việc đảo lô (Batch) hoặc đảo vị trí GoF; Dropping loại có chủ đích một frame trong mỗi khối l nhằm giảm tải và khảo sát độ bền pipeline; Splitting tách stream thành nhiều nhánh dựa trên chỉ số mod, phục vụ mô phỏng phân nhánh dữ liệu, cân bằng tải và điều phối frame; còn Merging ghép lại các nhánh theo cơ chế xen kẽ vòng tròn, tái lập một stream thống nhất. Mỗi phép toán biến đổi stream theo một cách riêng, tạo ra nhiều dạng stream đầu vào pipeline giúp đánh giá độ ổn định và hành vi của pipeline.

Về mặt quan hệ, bốn phép toán tạo thành một biến đổi phụ thuộc lẫn nhau. Reversing đảo lại thứ tự trong từng khối, làm đầu vào cho Dropping xác định phần tử bị loại. Splitting sau đó tách stream dựa trên chỉ số đã thay đổi qua hai bước trước, nên dạng các nhánh splitting là hệ quả trực tiếp của Reversing và Dropping. Cuối cùng, Merging phụ thuộc vào Splitting vì chỉ hoạt động khi có nhiều nhánh, và trật tự ghép cũng phản ánh cách Splitting phân bố frame theo mod. Các phép toán Reversing → Dropping → Splitting → Merging tạo nên một pipeline biến đổi stream liên hoàn, cho phép mô phỏng các tình huống thực tế như đảo frame, giảm mẫu, phân luồng song song và ghép xen kẽ vòng tròn để đánh giá sự ổn định và thông lượng của hệ thống.

e) Dữ liệu cho các tham số cấu hình của các kịch bản

Luận án cung cấp dữ liệu cho các tham số cấu hình tùy ý theo bảng 2.23. Môi trường giả lập: colab.research.google.com và ngôn ngữ lập trình Python.

Bảng 2.23. Dữ liệu của các tham số cấu hình cho từng kịch bản

Kịch bản	Các tham số cấu hình cho 6 phép toán xử lý stream												
	Take và Zip							*Rev*	*Drop*		*Split*		*Merge*
	T	M	fps_{in}	l	I_{keep}	Gof_{aware}	P	l	l	k	l	k	Use
KB1	5	1	[60]	4	[0,1,2,3]	False	Skip	5	8	2	8	[0,2]	False
KB2	5	2	[60,45]	4	[1,3]	False	Skip	4	4	2	4	[0,2]	True
KB3	5	3	[60,45,30]	4	[1,3]	True	Skip	3	3	2	3	[0,2]	True
KB4	5	4	[60, 45, 30, 20]	4	[0,2]	False	Skip	5	5	2	5	[0,2]	True
KB5	5	5	[60, 45, 30, 20, 15]	4	[1,3]	False	Skip	6	6	2	6	[0,2]	True
KB6	5	5	[60, 30, 45, 15, 10]	4	[1,3]	False	Skip	7	7	2	7	[0,2]	True

f) Chạy mô phỏng và thu thập kết quả KPIs

Lấy dữ liệu từ các tham số cấu hình ở bảng 2.23 để chạy mô phỏng và thu thập kết quả lượng hóa các chỉ số hiệu năng chính (KPIs). Kết quả mô phỏng của 6 kịch bản được trình bày chi tiết ở bảng 2.24 sau đây:

Bảng 2.24. Các KPIs so sánh giữa các KPIs của các kịch bản khi chạy mô phỏng

Kịch bản	Các chỉ số hiệu năng (KPIs) được lượng hóa dựa vào tham số cấu hình						
	TP	LM	LP	BA	BP	FI	IO
KB1	60.0	35.0000	35.0	2.0949	3.0	1.0000	0.0

KB2	52.4	36.2023	40.0	1.8921	5.0	0.9794	0.5
KB3	77.4	37.2739	45.0	2.8551	5.0	0.9276	0.5
KB4	65.0	38.3692	45.0	2.3926	5.0	0.7958	0.5
KB5	84.8	40.0825	50.0	3.2927	6.0	0.8073	0.5
KB6	79.8	38.7093	45.0	3.0200	6.0	0.7463	0.5

Phân tích kết quả KPIs của bảng 2.24 cho thấy rằng: Các chỉ số hiệu năng (KPIs) phản ánh rõ tác động từng chính sách cho taking, zipping, reversing, dropping và splitting lên hiệu quả vận hành của pipeline. Cụ thể, đặt 5 *câu hỏi/trả lời* để phân tích sự thay đổi trong cấu hình mô phỏng làm thay đổi đáng kể TP, LM, LP cũng như BA và BP như sau:

Câu hỏi 1: Kịch bản (KB) nào đạt thông lượng (TP) cao nhất hoặc thấp nhất?

- TP cao nhất: KB5 = 84.8 (M=5, nhiều nguồn nên dẫn đến tổng thông lượng lớn. Do BA = 3.2927 và BP = 6.0 giúp pipeline hoạt động trơn tru, ít nghẽn).
- TP thấp nhất: KB2 = 52.4 (Phép toán taking giữ một nửa nên dẫn đến IO = 0.5. Do BA = 1.8921 nhỏ nhất suy ra bộ đệm hạn chế, gây giảm thông lượng).

Câu hỏi 2: Độ trễ (LM, LP) ở KB nào tốt nhất hoặc xấu nhất?

- Độ trễ tốt nhất: KB1 có LM = 35.0 ms, LP = 35.0 ms, phản ánh đặc trưng pipeline chuẩn, không chịu tác động của các chính sách giảm tải hoặc tái cấu trúc.
- Độ trễ xấu nhất: KB5 có LM = 40.0825 ms, LP = 50.0 ms, vì do KB5 có 5 nguồn dẫn đến tranh chấp lịch, độ trễ bách phân vị thứ 95 bị đội lên. Điều này là hợp lý vì KB5 có BA và BP cao, cộng với IO = 0.5. Do tập hợp lưu lượng từ ba nguồn làm gia tăng cạnh tranh trong lịch biểu cũng như khiến hàng đợi tại các giai đoạn tích lũy nhiều hơn.

Câu hỏi 3: KB nào tiết kiệm bộ đệm nhất và KB nào tốn bộ đệm nhất (so sánh BA-BP)?

- Nếu xét tổng mức sử dụng bộ đệm (xấp xỉ BA·BP)
  - Tiết kiệm nhất: KB1 với BA = 2.0949, BP = 3.0 → BA·BP ≈ 6.28 (nhỏ nhất). Về giá trị BA riêng lẻ, KB2 có BA = 1.8921 nhỏ hơn, nhưng phải duy trì bộ đệm ở 5 stage (BP = 5.0), nên tổng bộ đệm vẫn lớn hơn KB1.
  - Tốn bộ đệm nhất: KB5 với BA = 3.2927, BP = 6.0 → BA·BP ≈ 19.76, cao nhất do phải gộp lưu lượng từ 3 nguồn và nhiều stage cần bộ đệm.
- Tham chiếu: KB1 (BA = 2.0949, BP = 3.0) có thể xem như pipeline chuẩn (không giảm tải, một nguồn), dùng làm mốc so sánh. Các kịch bản còn lại (KB2–KB6) bật thêm các chính sách taking, splitting hoặc đa nguồn nên tổng bộ đệm cần thiết đều cao hơn KB1.

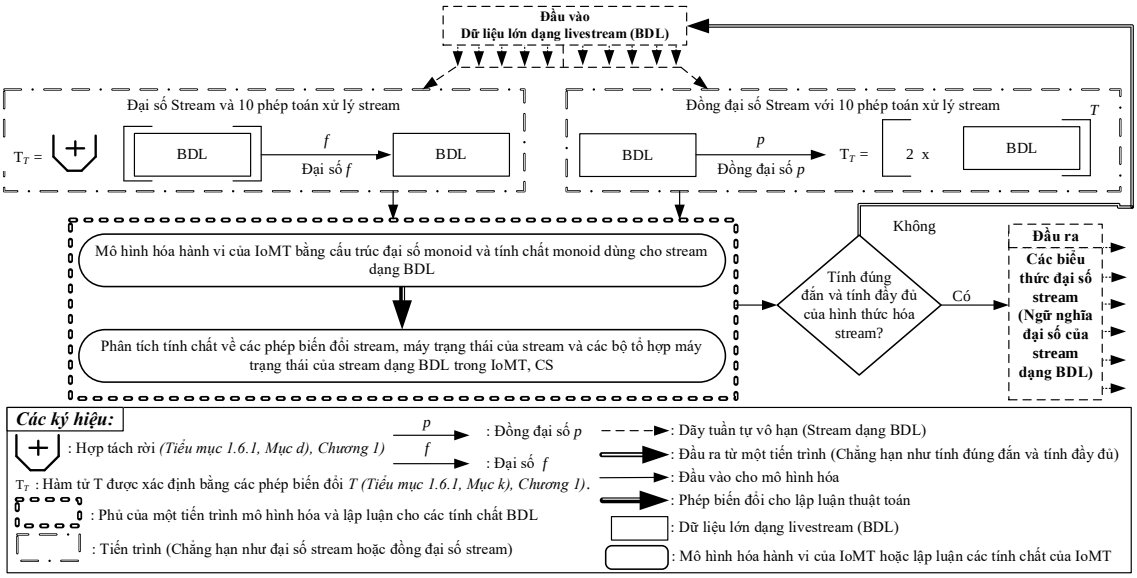
Câu hỏi 4: Cân bằng nguồn (FI) khác nhau thế nào giữa KB một nguồn và nhiều nguồn?

- Kịch bản một nguồn (KB1–KB4, KB6): Chỉ có KB1 đạt FI = 1.0000, nghĩa là cân bằng tuyệt đối giữa các frame trong GOF; Các kịch bản còn lại đều FI < 1, cho thấy mức độ mất cân bằng tăng dần: KB2 = 0.9794 (Gần cân bằng), KB3 = 0.9276 (Lệch nhẹ), KB4 = 0.7958 (Lệch đáng kể), KB6 = 0.7463 (Lệch nặng nhất trong nhóm một nguồn). Như vậy, Dù cùng là "Một nguồn", chỉ KB1 giữ được sự phân bổ hoàn hảo, còn các phiên bản bật thêm chính sách xử lý (Taking, Splitting, Skipping…) đều giảm FI.
- Kịch bản nhiều nguồn (KB5): FI = 0.8073, thấp hơn KB2–KB3 và chỉ nhỉnh hơn KB4, KB6. Điều này cho thấy việc trộn (Merging) nhiều nguồn gây mất cân bằng tự nhiên giữa các frame vì tranh chấp tài nguyên và lịch biểu. KB5 không phải FI thấp nhất bảng, nhưng việc gộp nguồn làm FI vẫn giảm rõ rệt so với KB1–KB2.

Câu hỏi 5: Các cấu hình(Gofaware bật/tắt, bỏ I-frame, tình huống đến sớm) ảnh hưởng gì đến TP/BA/LM?

- Gof_{aware} bật KB3 so với tắt KB2 và cả hai có cùng $l = 4$, $I_{keep} = [1,3]$, Skip: TP tăng 52.4 lên 77.4 (Gof_{aware} bật giúp giữ lại nhiều frame hiệu dụng nhưng bộ đệm lớn lên), LM tăng từ 36.2023 lên 37.2739, BA tăng 1.8921 lên 2.8551 (Bộ đệm tăng làm trễ tổng tăng một phần).
- KB4 là kịch bản bỏ I-frame (I-frame skipping), ta so với KB1 làm chuẩn: TP tăng nhẹ từ 60.0 lên 65.0 (Việc cắt bỏ I-frame làm pipeline nhận nhiều P/B frame hơn làm tăng TP); BA tăng: 2.0949 lên 2.3926 (Vì pipeline phải giữ nhiều frame liên tục hơn khi không còn I-frame làm điểm reset); LM tăng từ 35 lên 38.37 (Không có I-frame khiến chu kỳ đồng bộ kéo dài làm tăng trễ). Do đó, bỏ I-frame cải thiện TP nhưng làm LM và BA tăng đáng kể.
- KB6 mô phỏng trường hợp frame đến: TP tăng rất mạnh từ 60.0 lên 79.8 (Đến sớm giúp pipeline có sẵn dữ liệu ở đầu vào); BA tăng mạnh từ 2.0949 lên 3.0200 (Vì pipeline phải giữ thêm dữ liệu đến sớm dẫn đến sử dụng buffer lớn hơn); LM tăng từ 35 lên 38.70 (Lượng đệm tăng dẫn đến độ trễ nội bộ tăng theo). Như vậy, đến sớm giúp pipeline không bị thiếu nên TP rất cao, nhưng đổi lại BA và LM đều tăng.
- Ảnh hưởng của IO: KB1 không áp dụng taking, giữ nguyên toàn bộ frame nên dẫn đến IO=0.0; Tất cả KB2–KB6 áp dụng taking nên IO = 0.5, nghĩa là pipeline chỉ giữ lại 50% frame đầu vào (Điều này giúp giảm tải nhưng có thể làm TP giảm). Do đó, taking (IO=0.5) giúp pipeline giảm tải đầu vào nhưng không phải lúc nào cũng giúp cải thiện TP; tuỳ vào BA, BP, và cách frame được giữ lại.

2.4. Khung hình thức hóa dùng cho stream dạng BDL trong IoMT



Hình 2.11. Khung hình thức hóa dùng cho stream dạng BDL trong IoMT

Hình thức hóa stream dạng BDL gặp nhiều thách thức trong đại số stream và đồng đại số stream, chương này xây dựng một số khía cạnh đại số để hình thức hóa cơ chế hoạt động của stream dạng BDL trong IoMT [CT.1][CT.2][CT.3][CT.5] [CT.7][33]. Tính toán stream để hình thức hóa stream dạng BDL trong IoMT là một cấu xạ $IoMT^{\omega} \rightarrow IoMT^{\omega}$ *(Tiểu mục 1.6.1, Mục i), Chương 1)* tạo ra pipeline *(Tiểu mục 1.5, Chương 1)* đáp ứng ngữ nghĩa đại số của stream dạng BDL. Việc hình thức hóa như vậy được thể hiện qua một khung hình thức hóa cho stream dạng BDL trong IoMT ở hình 2.11.

Khung này có đầu vào là stream dạng BDL, đại số stream và các phép toán xử lý stream *(Tiểu mục 2.3.1)*, và đồng đại số stream *(Tiểu mục 2.3.5)* được áp dụng để biểu diễn cơ chế hoạt động của stream dạng BDL, kế tiếp việc mô hình hóa hành vi của CS cần có cấu trúc monoid *(Tiểu mục 1.6.1, Mục g), Chương 1)*, các phép biến đổi stream dạng BDL, máy trạng thái stream dạng BDL, và các bộ tổ hợp máy trạng thái stream dạng BDL. Cuối cùng, nếu khung này có đáp ứng tính đúng đắn và tính đầy đủ của hình thức hóa stream thì đầu ra là ngữ nghĩa đại số của stream dạng BDL, ngược lại BDL tiếp tục được đưa vào khung để hình thức hóa stream [CT.3].

Hình 2.11 mô tả đồng đại số stream với 10 phép toán xử lý stream (Dựa vào định nghĩa 2.15) được ứng dụng vào stream dạng BDL như sau: Một tập BDL của các trạng thái và tập các phép toán xử lý stream T, nếu cả BDL và T là hữu hạn, khi đó ta có một tiến trình xử lý stream hữu hạn, ngược lại ta có một tiến trình xử lý stream vô hạn. Một cặp hàm $\langle o_{\text{BDL}}, e_{\text{BDL}} \rangle$ có thể được mô tả chi tiết như sau:

- Tập phép toán xử lý stream dạng BDL $T \to 2$ được ký hiệu 2^T và $T \to$ BDL được ký hiệu BDL^T.
- Nói một cách khác, $2^T = \{f \mid f: T \to 2\}$ và $\text{BDL}^T = \{g \mid g: T \to \text{BDL}\}$. Do đó, cặp hàm $\langle o_{\text{BDL}}, e_{\text{BDL}} \rangle$ sẽ trở thành $\langle o_{\text{BDL}}, e_{\text{BDL}} \rangle$: $\text{BDL} \to (2 \text{ x BDL})^T$.

2.5. Cấu trúc monoid của BDL trong IoMT

2.5.1. Cấu trúc monoid dùng cho stream dạng BDL trong IoMT

Sử dụng phép toán stream để xử lý stream dạng BDL áp dụng vào cấu trúc đại số monoid là một tiếp cận nền tảng. Luận án ký hiệu $\circledast$ là đại diện cho các phép toán hai ngôi và ký hiệu $\boxdot$ là đại diện cho các phép toán một ngôi trong công trình [CT.2]. Phép toán stream này tạo ra tiến trình tự động mô hình hóa stream đầu vào từng frame tuần tự mà nó không được cung cấp đầy đủ từ khi nó bắt đầu vào CS.

Dựa vào phân tích BDL, với phép tính stream và thứ tự của các frame đến, và bộ đếm các frame (Counter of frames) cũng được gọi là tính toán stream. Luận án minh họa một đặc tả hoạt động cho bộ đếm các frame là mỗi khi một frame mới xuất hiện, bộ đếm từng frame sẽ phát sinh tổng số các frame cho đến thời điểm này, và biểu diễn hoạt động bằng một hàm toán học β sau đây (2.22) [CT.3]:

- $\beta(\langle x_1, x_2, x_3, \ldots, x_n \rangle) = \langle 1, 2, 3, \ldots, n \rangle$ (2.22)

Hàm β (2.22) có dãy tuần tự các frame có dạng $\langle x_1, x_2, \ldots, x_n \rangle$ là đầu vào và $\langle 1, 2, \ldots, n \rangle$ là kết quả đầu ra được tích luỹ của phép tính stream. Với tính toán stream như vậy, đầu vào có thể được đặc tả một cách rõ ràng dưới dạng nhiều bộ thay vì nguyên cả stream. Như vậy, nhiều bộ cũng có thể được coi như là stream dạng BDL.

Ví dụ 2.15: Minh họa đầu vào của bộ đếm frame là dãy các frame $\langle x_1, x_2, x_3 \rangle = \langle 1,1,0 \rangle$, suy ra kết quả là $\langle 1,1,0 \rangle$, đầu vào của bộ đếm frame tiếp theo là $\langle x_1, x_2, x_3 \rangle =$

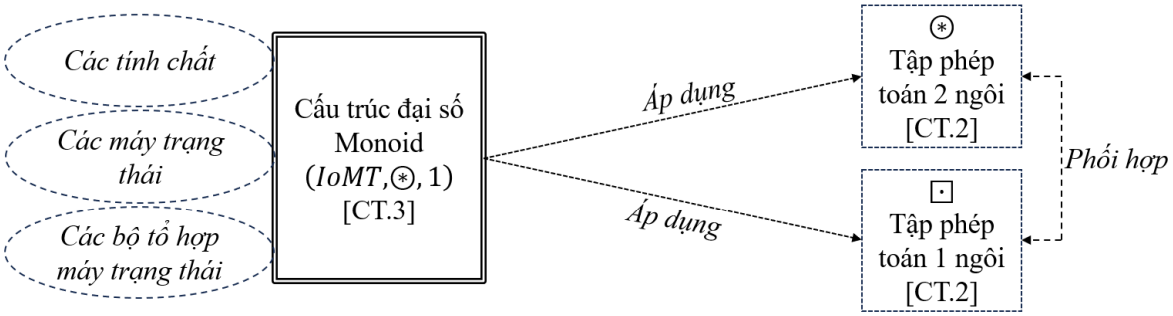
⟨0,1,1⟩, suy ra kết quả là ⟨1,2,1⟩, và cứ như vậy.

Luận án nhận thấy có ba khái niệm tách biệt sau đây cần được mô hình hóa toán học cho stream dạng BDL được xem xét tại thời điểm bắt đầu tính toán stream, một stream gồm:

- *Các frame dạng BDL trong IoMT*
- *Mở rộng BDL trong IoMT bằng các frame được bổ sung*
- *Các BDL trong IoMT rỗng*

Từ đây giúp luận án xây dựng một cấu trúc monoid được xem như là một cấu trúc đại số đặc biệt thỏa mãn cho stream. BDL cần được mô hình hóa bằng các frame thông qua tính toán stream để nối các frame và một phần tử đơn vị tách biệt 1 đồng nhất dùng để đặc tả frame rỗng.

Dựa vào Định nghĩa 1.1 về cấu trúc monoid *(Tiểu mục 1.6.1, Mục g), Chương 1)*, mười phép toán xử lý stream (⊡ hoặc ⊛) (ở trang 11 trong công trình [CT.2]), luận án đề xuất xây dựng cấu trúc đại số monoid dùng cho stream dạng BDL trong IoMT (Hình 2.12). Đây là lớp các cấu trúc đại số thỏa mãn mô hình hóa các stream hoặc các frame dạng BDL qua các tính chất, máy trạng thái và các bộ tổ hợp máy trạng thái (Hình 2.12).



Hình 2.12. Mô tả cấu trúc đại số monoid áp dụng các phép toán stream

Định nghĩa 2.16 (Cấu trúc monoid): Một monoid là một bộ ba $(IoMT, \circledast, 1)$, trong đó: $IoMT$ là một tập hợp các trạng thái hữu hạn hoặc kiểu các stream dữ liệu, cụ thể là BDL. Khi đó, $\circledast$: $IoMT \times IoMT \to IoMT$ là một phép toán hai ngôi trên $IoMT$, gọi là phép nối các frame trong ngữ cảnh stream, và $1 \in IoMT$ là phần tử đơn vị, sao cho hai tiên đề (2.23) và (2.24) sau được thỏa mãn [CT.3]:

- *Tính chất kết hợp:* $(x \circledast y) \circledast z = x \circledast (y \circledast z), \forall x, y, z \in IoMT$ (2.23)
- *Tính chất đồng nhất:* $1 \circledast x = x \circledast 1 = x, \forall x \in IoMT$ (2.24)

Tiên đề (2.23) đặc tả $\circledast$ là sự kết hợp các stream dạng BDL và tiên đề (2.24) đặc tả 1 là đơn vị trái và đơn vị phải cho phép toán stream $\circledast$. Ngoài ra, đôi khi đặc tả xy cũng ám chỉ cho biểu diễn $x \circledast y$. Giả sử $IoMT$ là một monoid, đặc tả $IoMT^*$ cho bộ các frame BDL hữu hạn của $IoMT$ và ε là bộ frame BDL rỗng.

Định nghĩa 2.17 (Tích của hai monoid): Cho $(IoMT_1, \circledast_{IoMT_1}, 1_{IoMT_1})$ và

$(IoMT_2, \circledast_{IoMT_2}, 1_{IoMT_2})$ là hai cấu trúc đại số monoid. Khi đó, tích của hai monoid này là monoid $(IoMT_1 \times IoMT_2, \circledast, 1)$, trong đó: phép toán stream $\circledast$ được cho bởi: $(x, y) \circledast (x', y') = (x \circledast_{IoMT_1} x', y \circledast_{IoMT_2} y')$, $\forall x, x' \in IoMT_1$ và $\forall y, y' \in IoMT_2$, và phần tử đơn vị 1 là $1 = (1_{IoMT_1}, 1_{IoMT_2})$ [CT.3].

Định nghĩa 2.18 (Đồng cấu monoid cho stream): Cho $(IoMT_1, \circledast, 1)$ và $(IoMT_2, \circledast, 1)$ là hai cấu trúc monoid. Một ánh xạ $h: IoMT_1 \to IoMT_2$ được gọi là một đồng cấu monoid nếu thỏa mãn hai điều kiện sau: Bảo toàn phần tử đơn vị: $h(1) = 1$ và bảo toàn phép toán stream: $h(x \circledast y) = h(x) \circledast h(y)$, $\forall x, y \in IoMT_1$ và $IoMT_2$ [CT.3].

Định nghĩa 2.18 ở trên thể hiện tính chất đảm bảo rằng hàm h bảo toàn phép toán stream và phần tử đơn vị, do đó phù hợp để mô hình hóa sự chuyển đổi stream dữ liệu một cách bảo toàn cấu trúc.

Nhận thấy, ý nghĩa đồng cấu monoid này có thể mô hình hóa quá trình xử lý dữ liệu video theo stream một cách song song và bảo toàn cấu trúc thời gian. Điều này rất quan trọng trong: hệ thống giám sát thời gian thực, phân tích hành vi trong video, các đường ống (pipeline) xử lý video phân tán trong các hệ thống xử lý dữ liệu lớn như: Apache Flink, Spark Streaming (Bảng 2.25).

Luận án mô phỏng pipeline không nhằm thay thế các hệ thống thực tế như Apache Flink hay Spark Streaming, mà đóng vai trò là mô hình lý thuyết để phân tích và kiểm chứng hình thức các tính chất như độ trễ, thông lượng và tính đúng đắn của quá trình tổng hợp. Trong khi Apache Flink và Spark Streaming mạnh về triển khai và tối ưu thực nghiệm, chúng không cung cấp cơ chế kiểm chứng hình thức ở mức ngữ nghĩa như mô hình đại số stream của luận án.

Bảng 2.25. So sánh mô phỏng pipeline và hệ thống streaming thực tế

Tiêu chí	Mô phỏng pipeline	Apache Flink	Spark Streaming
Mục tiêu	Phân tích và kiểm chứng hình thức	Xử lý stream thời gian thực	Xử lý theo các lô nhỏ theo thời gian
Mô hình	Đại số stream	Bộ máy xử lý theo luồng dữ liệu	Lô mở rộng
Phân tích độ trễ	Phân tích và chứng minh	Đo đạc thực nghiệm	Đo đạc
Kiểm chứng đúng đắn	Có (Đồng quy nạp)	Không	Không
Vai trò	Cơ sở lý thuyết	Triển khai thực tế	Triển khai thực tế

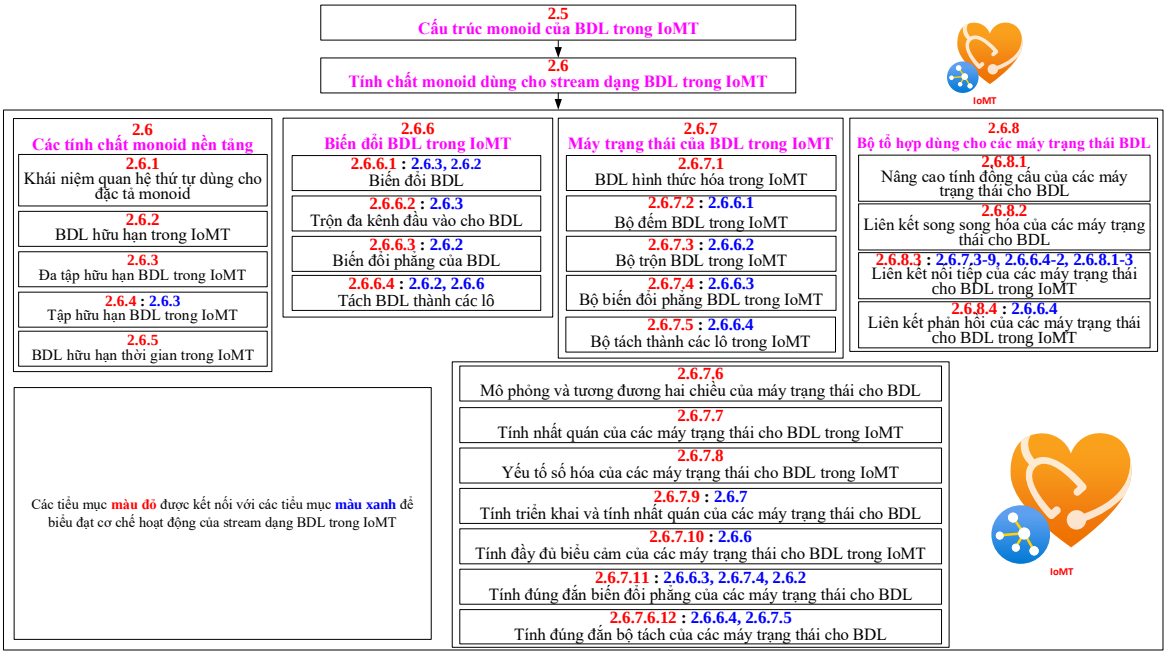
Ví dụ 2.16: Đồng cấu monoid trong BDL. Giả sử một hệ thống phân tích video livestream từ camera giám sát sử dụng cấu trúc monoid để mô tả stream dữ liệu: Mỗi luồng video được xem là một dãy các frame: $x = [f_1, f_2, \ldots f_n] \in SFrame$, trong đó

mỗi f_i là một frame rút trích từ video tại một thời điểm cụ thể. Phép toán $\circledast$ là nối stream theo thời gian thực, tức là kết hợp tuần tự các frame: $x \circledast y = x + y$, trong đó $+$ là phép toán merging hai stream. Phần tử đơn vị 1 là stream rỗng, tức là stream không chứa frame nào. Khi đó, $(SFrame, \circledast, 1)$ là một cấu trúc monoid.

2.5.2. Hình thức hóa stream dạng BDL trong IoMT

Bảng 2.26. Mô tả tổng hợp tính chất monoid dùng cho stream dạng BDL

Nhóm các nội dung	Mô tả tóm tắt
Các tính chất monoid áp dụng cho BDL trong IoMT	▪ Quan hệ thứ tự của frame/stream/state trong IoMT thỏa mãn đặc tả monoid *(Tiểu mục 2.6.1)*. ▪ Tính chất monoid nền tảng cho trạng thái hữu hạn của máy trạng thái BDL *(Tiểu mục 2.6.2–2.6.5)*.
Biến đổi BDL trong IoMT *(Tiểu mục 2.6.6)*	▪ Biến đổi từ tập trạng thái $IoMT_1$ sang $IoMT_2$ *(Tiểu mục 2.6.6)*.
Máy trạng thái BDL trong IoMT *(Tiểu mục 2.6.7)*	▪ Xây dựng các máy trạng thái: Hình thức hóa; Bộ đếm BDL; Bộ trộn BDL; Bộ biến đổi phẳng BDL; Bộ tách BDL thành các lô.
Tổ hợp máy trạng thái *(Tiểu mục 2.6.8)*	▪ *Song song:* Từng liên kết đại diện phép toán độc lập. ▪ *Nối tiếp:* Đầu ra của máy thứ nhất là đầu vào của máy thứ hai. ▪ *Phản hồi:* Luồng tính toán trong vòng lặp được xác định rõ để tránh phân kỳ.
Lợi ích của tính chất monoid trong IoMT	▪ Xử lý stream dữ liệu dạng BDL theo thời gian thực. ▪ Duy trì livestream ổn định ngay cả khi có lỗi dữ liệu. ▪ Giảm tải băng thông và tối ưu mức xử lý. ▪ Đảm bảo tính toàn vẹn và bảo mật dữ liệu. ▪ Tăng cường hiệu suất AI trong phân tích dữ liệu IoMT.



Hình 2.13. Sơ đồ kết nối cấu trúc monoid và các tính chất monoid dùng cho stream

Luận án mô tả mối quan hệ của cấu trúc đại số monoid và các tính chất monoid nhằm để hình thức hóa (mô hình hóa, phân tích tính chất, kiểm chứng hình thức) cơ chế hoạt động của stream dạng BDL trong IoMT bằng sơ đồ ở hình 2.13 và bảng 2.26 mô tả tổng hợp tính chất monoid dùng cho stream dạng BDL. Tính chất monoid giúp ta hiểu được cơ chế hoạt động (vào, hoạt động, ra) của stream trong các hệ thống tính toán (IoT, IoMT). Các tiểu mục 2.6.1-2.6.8 nhấn mạnh sự đa dạng của các đối tượng toán học, có thể được coi là một cách tiếp cận để diễn đạt ý nghĩa cơ chế hoạt động của stream dạng BDL trong IoT, IoMT (Hình 2.13).

2.6. Tính chất monoid dùng cho stream dạng BDL trong IoMT

2.6.1. Khái niệm quan hệ thứ tự dùng cho đặc tả monoid

$\forall x, y \in IoMT$, $x \preccurlyeq y$ nếu tồn tại $z \in IoMT$ sao cho $xz = y$. Quan hệ $\preccurlyeq$ này được đặc tả thứ tự trước cho $(IoMT, \circledast, 1)$, phần tử đơn vị 1 liên quan đến $1 \circledast x = x$ ám chỉ $1 \preccurlyeq x$, $\forall x \in IoMT$ [CT.3].

Hàm prefix: $IoMT \times IoMT \to \mathcal{P}(IoMT)$ gọi là hàm tiền tố, được định nghĩa $\text{prefix}(x, y) = \{z \in IoMT \mid xz = y\}$, $\forall x, y \in IoMT$. Quan hệ $x \preccurlyeq y$ tồn tại khi và chỉ khi $\text{prefix}(x, y) \neq \emptyset$, và $\text{prefix}(x, y)$ thu thập tất cả nhân chứng cho $x \preccurlyeq y$.

Hàm $\delta: IoMT \times IoMT \to IoMT$ là hàm chứng tiền tố nếu miền xác định thỏa $\preccurlyeq$ và thỏa mãn $\delta(x, y) \in \text{prefix}(x, y)$, $\forall x, y \in IoMT$. Quan hệ $\preccurlyeq$ được biểu diễn qua tích của hàm prefix trên $\preccurlyeq$ như sau: $\prod_{(x, y) \in \preccurlyeq} \text{prefix}(x, y)$.

Cấu trúc $(IoMT, \circledast, 1)$ có tính chất rút gọn trái nếu $xy = xz$ suy ra $y = z$, $\forall x, y, z \in IoMT$. Do đó, $IoMT$ có tính chất rút gọn trái và hàm chứng tiền tố tách rời, đảm bảo tồn tại z tách rời để $xz = y$.

2.6.2. BDL hữu hạn trong IoMT

Xét $(\text{FBDL}(IoMT), \circledast, \varepsilon)$, $\text{FBDL}(IoMT)$ biểu diễn tập frame BDL hữu hạn trong $IoMT$, $\circledast$ là nối frame và ε là frame rỗng, $\forall x, y \in \text{FBDL}(IoMT)$.

$$\frac{(\text{FBDL}(IoMT), \circledast, \varepsilon) \text{ và } \forall x, y \in \text{FBDL}(IoMT)}{\exists z \in \text{FBDL}(IoMT) \text{ sao cho } xz = y}$$

Xét một biến thể để làm sáng tỏ việc liên quan đến quan hệ $\preccurlyeq$ cho tập frame BDL hữu hạn $IoMT^*$, $\circledast$ được dùng cho định nghĩa $x \circledast y = xy$ và ε là frame rỗng.

$$\frac{(IoMT^*, \circledast, \varepsilon) \text{ và } x \preccurlyeq y}{\text{Frame } y \text{ là hậu tố của frame } x}$$

2.6.3. Tập tất cả đa tập hữu hạn BDL trong IoMT

$$\frac{(\text{FMultiset}(IoMT), \cup, \emptyset), \quad x, y \in \text{FMultiset}(IoMT), x \preccurlyeq y \text{ và } x \subseteq y}{\exists z \in \text{FMultiset}(IoMT) \text{ sao cho } x \cup z = y}$$

Trong đó:

- $\text{FMultiset}(IoMT)$ là tập tất cả đa tập hữu hạn BDL trong $IoMT$,

- ∪ là hợp của đa tập,
- ∅ là tập của đa tập rỗng,
- (FMultiset($IoMT$), ∪, ∅) là một cấu trúc đại số monoid trong $IoMT$ có tính chất giao hoán với bộ sinh của $IoMT$ và có tính chất rút gọn trái.

2.6.4. Tập hữu hạn BDL trong IoMT

Xét (FSet($IoMT$), ∪, ∅), FSet($IoMT$) là tập các frame BDL hữu hạn của $IoMT$, ∪ là hợp của các tập hữu hạn, và ∅ là tập rỗng.

$$\frac{(\text{FSet}(IoMT),\cup,\ \emptyset) \text{ và } x \subseteq y}{\text{FSet}(IoMT)}$$

$\forall x, y \in \text{FMultiset}(IoMT)$, $x \preccurlyeq y$ nếu và chỉ nếu $x \subseteq y$. Với $x \subseteq y$, định nghĩa $\partial(x, y) = y \backslash x$, $\backslash$ là phép toán hiệu giữa các tập hữu hạn, với $x \cup (y \backslash x) = y$ suy ra $x \subseteq y$. Tương tự như vậy, định nghĩa $\tau(x, y) = y$ cho $x \subseteq y$. Do đó, $x \cup y = y$ suy ra $x \subseteq y$, τ cũng là một PWF. Do đó, FSet($IoMT$) có một số PWF.

2.6.5. BDL hữu hạn thời gian trong IoMT

$$\frac{(\text{TFBDL}(IoMT),\ \Diamond,\ 0) \text{ và TFBDL là } s_0 \circ a_0 \circ s_1 \circ a_1 \circ \cdots \circ s_{n-1} \circ a_{n-1}}{\text{TFBDL}(IoMT) = \mathbb{N} \circ (\mathbb{N} \circ IoMT)^*}$$

Trong đó:

- (TFBDL($IoMT$), ◊, 0) là một monoid
- 0 là phần tử đơn vị BDL được định thời gian
- TFBDL là một BDL tuần tự xen kẽ $s_0 \circ a_0 \circ s_1 \circ a_1 \circ \cdots \circ s_{n-1} \circ a_{n-1}$
- $s_i \in \mathbb{N}$ là các dấu thời gian
- $a_i \in IoMT$ cho mỗi frame thứ i
- Các lần xuất hiện $s_0, s_1, s_2, \ldots$ được gọi là dấu thời gian biểu diễn cho thời gian đã trôi qua. Vì vậy, việc thu thập tất cả các BDL được định thời gian là bằng biểu thức $\text{TFBDL}(IoMT) = \mathbb{N} \circ (\mathbb{N} \circ IoMT)^*$.

Như vậy, phép toán tích trộn ◊ của stream dạng BDL được định dấu thời gian: $s_0 \circ a_0 \circ s_1 \circ a_1 \circ s_2 \circ a_2 \circ \cdots \circ s_{m-1} \circ a_{m-1} \Diamond t_0 \circ b_0 \circ t_1 \circ b_1 \circ t_2 \circ b_2 \circ \cdots \circ t_{n-1} \circ b_{n-1} \Diamond w_0 \circ c_0 \circ w_1 \circ c_1 \circ w_2 \circ c_2 \circ \cdots \circ w_{p-1} \circ c_{p-1} = s_0 \circ a_0 \circ s_1 \circ a_1 \circ s_2 \circ a_2 \circ \cdots \circ a_{m-1} \circ (s_m + t_0) \circ b_0 \circ t_1 \circ b_1 \circ t_2 \circ b_2 \circ \cdots \circ b_{n-1} \circ (t_n + w_0) \circ c_0 \circ w_1 \circ c_1 \circ w_2 \circ c_2 \circ \cdots \circ w_{p-1} \circ c_{p-1}$.

Từ tiểu mục 2.6.1 đến 2.6.5, luận án đã nhấn mạnh sự đa dạng của các đối tượng toán học để diễn đạt ý nghĩa về cơ chế hoạt động của stream. Stream dạng BDL này có thể được tạo ra một cách đa dạng bằng cách sử dụng cấu trúc monoid. Stream của tiểu mục 2.6.2 là bộ các frame của tiểu mục 2.6.3 có thể được mô tả như là sự tương đương theo thứ tự của các tiểu mục tiếp theo và được đề xuất để đưa ra các khái niệm về stream.

2.6.6. Biến đổi BDL trong IoMT

Biến đổi BDL (BDLTrans) là coi BDL như một dãy tuần tự các frame. Biến đổi BDL là quá trình suy diễn từ dãy frame đầu vào sang dãy frame đầu ra, được đặc tả bằng máy trạng thái hữu hạn và các bộ nhận đầu ra mở rộng.

Các biến đổi BDL được phân loại dựa trên tính chất monoid của BDL đầu vào và đầu ra. Chúng được mô hình hóa bằng các hàm đơn điệu, ánh xạ lịch sử BDL đầu vào (các frame từ đầu đến hiện tại) sang lịch sử BDL đầu ra (các frame đã tạo ra). Tính đơn điệu đảm bảo rằng các kết quả đã phát ra là không thể thu hồi, thể hiện tính chất monoid của biến đổi BDL. Luận án mô tả các tính chất này như một mô hình ngữ nghĩa đại số để xử lý stream BDL trong IoMT.

Các tính chất monoid của biến đổi BDL được mô tả bằng các mô hình ngữ nghĩa toán học. Tính toán stream BDL sử dụng lịch sử đầu vào (BDL trong IoMT từ khi bắt đầu đến hiện tại) để tạo ra lịch sử đầu ra (các frame tích lũy đã phát).

Với phép tính stream BDL ánh xạ lịch sử đầu vào sang lịch sử đầu ra qua hàm đơn điệu $\beta: IoMT_1 \rightarrow IoMT_2$, ở đó có $(IoMT_{1,2}, \circledast, 1)$ biểu diễn đầu vào và đầu ra.

$$\frac{\beta: IoMT_1 \rightarrow IoMT_2, x \preccurlyeq y}{\beta(x) \preccurlyeq \beta(y), \forall x,\ y\ \in IoMT_1}$$

Với hàm β, mô tả μ là một MWF, nếu nó ánh xạ các frame $x,\ y \in IoMT_1$,

$$\frac{z \in \text{prefix}(x,\ y), x \preccurlyeq y}{\mu(x,\ y,\ z) \in \text{prefix}\big(\beta(x),\ \beta(y)\big), \beta(x) \preccurlyeq \beta(y)}$$

Được biểu đạt qua tích của hàm prefix trên $\preccurlyeq$ và yêu cầu μ phải thuộc:

$$\frac{\prod_{x,\ y \in IoMT_1} \text{prefix}(x,\ y) \rightarrow \text{prefix}(\beta(x),\ \beta(y))}{\beta(x) \circ \mu(x,\ y,\ z) = \beta(y) \text{ với } xz = y \text{ hoặc } \beta(x) \circ \mu(x,\ z) = \beta(xz), \forall x,\ z \in IoMT_1}$$

Việc mô tả mọi tiền tố giữa hai frame đầu vào x và y thuộc $IoMT_1$ sẽ được ánh xạ một cách đơn điệu sang các tiền tố tương ứng giữa hai frame đầu ra $\beta(x)$ và $\beta(y)$ thông qua hàm β. Đây là một đặc tả toán học cho sự bảo toàn tính tuần tự và tính tích lũy trong quá trình biến đổi stream thông qua hàm β. Vì vậy, để xác định tính chất của μ, $\forall x, y, z \in IoMT_1$ với $xz = y$ nó đúng đắn với $\beta(x) \circ \mu(x, y, z) = \beta(y)$. Đôi khi để ngắn gọn, chỉ viết $\mu(x, z)$ để ký hiệu cho $\mu(x, xz, z)$, sau đó đặc tả các tính chất của μ được viết như sau: $\beta(x) \circ \mu(x, z) = \beta(xz), \forall x, z \in IoMT_1$.

Tập hợp biến đổi stream từ IoMT_1 đến IoMT_2 là $\text{BDLTrans}(IoMT_1, IoMT_2)$ theo β. Một biến đổi stream $\langle \beta, \mu \rangle$ được xây dựng quy nạp qua hàm $\text{F}(\beta,\ \mu): IoMT_1^* \rightarrow IoMT_2^*$,

$$\frac{\begin{array}{c}\text{với } \text{F}(\beta,\ \mu)(\varepsilon) = \langle \beta(1) \rangle \text{ và } \text{F}(\beta,\ \mu)(\langle x_1, \ldots,\ x_n,\ x_{n+1} \rangle) \\ = \text{F}(\beta,\ \mu)(\langle x_1, \ldots,\ x_n \rangle) \circ \langle \mu(x_1 \circ \ \cdot\ \cdot\ \cdot \circ x_n,\ x_{n+1}) \rangle\end{array}}{\text{Sinh ra các frame } \langle x_1, \ldots, x_{n+1} \rangle \in IoMT_1^*, \pi\big(F(\beta,\mu)(\bar{x})\big) = \beta\big(\pi(\bar{x})\big), \forall \bar{x}\ \in\ IoMT_1^*}$$

Trong đó, biến đổi stream $\langle \beta,\ \mu \rangle$ với các frame đầu vào $x,\ y \in IoMT_1$ có mức tăng đầu ra $\mu(x, y)$ khi x mở rộng thành xy trong $(IoMT_2, \circledast, 1)$ đầu ra, μ không được xác định tách biệt bởi $\beta(x)$ và $\beta(xy)$, cung cấp thêm thông tin tính toán stream ngoài hàm β. Tuy nhiên, nếu $(IoMT_2, \circledast, 1)$ đầu ra có thể rút gọn trái, thì tồn tại hàm μ tách biệt thể hiện tính đơn điệu của hàm β.

Cho $\langle \beta, \mu \rangle$: $\text{BDLTrans}(IoMT_1, IoMT_2)$ là một biến đổi stream dạng BDL trong IoMT. Hình thành tăng dần $F(\beta, \mu)$ của biến đổi $\langle \beta, \mu \rangle$ mô tả biến đổi stream cho việc tăng đầu vào và tăng đầu ra được minh họa qua ví dụ 2.17 sau:

Ví dụ 2.17: Mô tả biến đổi stream dạng BDL tăng đầu vào và tăng đầu ra.

- $F(\beta,\mu)(\langle x_1\rangle) = \langle\beta(1), \mu(1,x_1)\rangle$ và $F(\beta,\mu)(\langle x_1,x_2\rangle) = \langle\beta(1), \mu(1,x_1), \mu(x_1,x_2)\rangle$.
- Tính chất chung hình thành tăng dần các frame dạng BDL trong IoMT là $\pi\big(F(\beta,\mu)(\bar{x})\big) = \beta\big(\pi(\bar{x})\big), \forall \bar{x} \in IoMT_1^*$:
  - $\pi\big(F(\beta,\mu)(\langle x_1,x_2,x_3,x_4\rangle)\big)$
    $= \beta(1) \circ \mu(1,x_1) \circ \mu(x_1,x_2) \circ \mu(x_1x_2,x_3) \circ \mu(x_1x_2x_3,x_4)$
    $= \beta(x_1) \circ \mu(x_1,x_2) \circ \mu(x_1x_2,x_3) \circ \mu(x_1x_2x_3,x_4)$
    $= \beta(x_1x_2) \circ \mu(x_1x_2,x_3) \circ \mu(x_1x_2x_3,x_4)$
    $= \beta(x_1x_2x_3) \circ \mu(x_1x_2x_3,x_4)$
    $= \beta(x_1x_2x_3x_4)$

2.6.6.1. Phép biến đổi dùng cho bộ đếm BDL trong IoMT

Tính toán stream BDL cho $(\text{FMultiset}(IoMT), \circledast, 1)$ đầu vào (tiểu mục 2.6.3) và $(\text{FBDL}(IoMT), \circledast, 1)$ đầu ra (tiểu mục 2.6.2) theo hai cách biểu diễn sau:

- *Phi hình thức*: Tính toán này không có đầu ra ban đầu và mỗi khi có frame mới đến sẽ trả số lượng các frame đã xem cho đến bây giờ.
- *Hình thức*:

$$\frac{(\text{FMultiset}(IoMT), \circledast, 1) \text{ và } (\text{FBDL}(IoMT), \circledast, 1), \text{ phép biến đổi BDL qua hàm } \beta: \text{FMultiset}(IoMT) \to \text{FBDL}(IoMT) \text{ với } \beta(\emptyset) = \varepsilon \text{ và } \beta(x) = \langle 1, 2, \ldots, |x|\rangle}{\mu(x, \emptyset) = \varepsilon \text{ và } \mu(x, y) = \langle |x|+1, |x|+2, |x|+3, \ldots, |x|+|y|\rangle \text{ khi } y \neq \emptyset}$$

Trong đó, hàm β được đặc tả bằng biểu thức sau $\beta(\emptyset) = \varepsilon$ và $\beta(x) = \langle 1, 2, \ldots, |x|\rangle$ với mọi frame $x \in \text{FMultiset}(IoMT) \neq \emptyset$, $|x|$ biểu diễn số lượng của các frame x, do vậy biến đổi stream của β là đơn điệu. Vì $\text{FBDL}(IoMT)$ có thể rút gọn trái, nên MWF được xác định: $\mu(x,\emptyset) = \varepsilon$ và $\mu(x,y) = \langle |x|+1, \ldots, |x|+|y|\rangle$ khi $y \neq \emptyset$.

2.6.6.2. Trộn đa kênh đầu vào dùng cho BDL trong IoMT

$$\frac{\begin{array}{c}\text{FMultiset}(IoMT, \circledast, 1), \\ \beta: \text{FMultiset}(IoMT) \times \text{FMultiset}(IoMT) \to \text{FMultiset}(IoMT), \beta(x, y) = x \cup y\end{array}}{\mu(\langle x_1,y_1\rangle, \langle x_2,y_2\rangle) = (x_2 \cup y_2) \setminus (x_1 \cup y_1) \text{ với mọi } x_1, y_1, x_2, y_2 \in \text{FMultiset}(IoMT)}$$

Trong đó, đặc tả tính chất biến đổi trộn đa kênh, với hai kênh đầu vào và một kênh đầu ra có kiểu $\text{FMultiset}(IoMT)$. Hàm chứng đơn điệu $\beta: \text{FMultiset}(IoMT) \times \text{FMultiset}(IoMT) \to \text{FMultiset}(IoMT)$, được cho bởi biểu thức $\beta(x,y) = x \cup y$ với đa tập x và y mô tả quá trình trộn đa kênh đầu vào tách biệt của hai stream dạng BDL con đầu vào. Trong ngữ cảnh của phép tính stream, mỗi khi có một frame mới đến, được hiểu là bất kể kênh nào, nó đều được chuyển đến kênh đầu ra. Vì $\text{FMultiset}(IoMT)$ có thể rút gọn trái, MWF μ được xác định: $\mu(\langle x_1,y_1\rangle, \langle x_2,y_2\rangle) = (x_2 \cup y_2) \setminus (x_1 \cup y_1)$ cho mọi $x_1, y_1, x_2, y_2 \in \text{FMultiset}(IoMT)$.

2.6.6.3. Biến đổi phẳng của BDL trong IoMT

$$\frac{(IoMT,\circledast,1),\beta:\text{FBDL}(IoMT)\to IoMT \text{ được định nghĩa bởi } \beta(\bar{x})=\pi(\bar{x}),\ \forall\bar{x}\in\text{FBDL}(IoMT) \text{ và } \mu \text{ xác định bởi } \mu(\bar{x},\ \bar{y})=\pi(\bar{y}),\ \forall\bar{x},\bar{y},\in\text{FBDL}(IoMT)}{\text{even}(IoMT)=\langle\beta,\ \mu\rangle\in\text{BDLTrans}(\text{FBDL}(IoMT),IoMT)}$$

Trong đó, $(IoMT,\circledast,1)$ là một monoid, hàm $\beta:\text{FBDL}(IoMT)\to IoMT$ được định nghĩa bởi $\beta(\bar{x})=\pi(\bar{x}),\ \forall\bar{x}\in\text{FBDL}(IoMT)$, mô tả tính chất biến đổi phẳng cho các stream được sử dụng tạm thời để giảm frame cần truyền tải. Hàm β là đơn điệu và MWF μ của β được xác định bởi $\mu(\bar{x},\bar{y})=\pi(\bar{y}),\ \forall\bar{x},\bar{y},\in\text{FBDL}(IoMT)$ thì biến đổi phẳng $\text{even}(IoMT)=\langle\beta,\mu\rangle$ của BDL trong IoMT có dạng $\langle\beta,\mu\rangle\in\text{BDLTrans}(\text{FBDL}(IoMT),IoMT)$.

2.6.6.4. Tách BDL thành các lô trong IoMT

Cho tập hợp $\Sigma=\{a,b\}$ là tập hợp các frame. Giả sử muốn phân rã một BDL trong Σ^* thành các lô (batch) có kích thước 5 frame, để phân rã bằng cách sử dụng hai hàm $r_1:\Sigma^*\to\text{FBDL}(\Sigma^*)$ và $r_2:\Sigma^*\to\Sigma^*$. Biểu diễn phi hình thức, r_1 cung cấp dãy tuần tự các lô có khối lượng là 5 frame, và r_2 cung cấp phần còn lại của các lô frame chưa hoàn chỉnh.

Ví dụ 2.18: Tách BDL thành các lô frame: $r_1(abbbbaaaabba)=\langle abbbb,aaaab\rangle$ và $r_2(abbbbaaaabba)=\langle ba\rangle$.

$$\frac{(\Sigma^*,\circledast,1)\to(IoMT,\circledast,1)}{r=(r_1,r_2),\text{với } r_1:IoMT\to\text{FBDL}(IoMT) \text{ và } r_2:IoMT\to IoMT}$$

Trong đó, việc tách thành các lô như vậy có thể được tổng quát hóa từ $(\Sigma^*,\circledast,1)$ sang $(IoMT,\circledast,1)$. Do vậy, việc đặc tả một bộ tách các frame là một cặp hàm $r=(r_1,r_2)$, với $r_1:IoMT\to\text{FBDL}(IoMT)$ và $r_2:IoMT\to IoMT$ thỏa mãn các tính chất (2.25-2.28) tách frame thành các lô như sau:

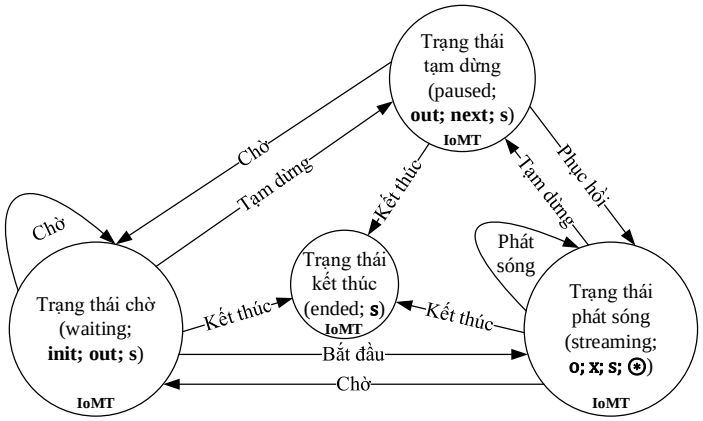
- $x=\pi(r_1(x))\circledast r_2(x)$ mô tả rằng r_1 và r_2 phân rã $x\in IoMT$, (2.25)
- $r_1(1_{IoMT})=1_{IoMT}$ mô tả rằng phần đơn vị không được phân rã, (2.26)
- $r_1(x\circledast y)=r_1(x)\circledast r_1(r_2(x)\circledast y)$ và (2.27)
- $r_2(x\circledast y)=r_2(r_2(x)\circledast y)$ (2.28)

Tính chất (2.25) và (2.26) ám chỉ rằng $r_2(1_{IoMT})=1_{IoMT}$. Tính chất (2.27) ám chỉ rằng r_1 là đơn điệu. Tính chất (2.27) và (2.28) mô tả cách phân rã phần ghép hai frame của monoid. Định nghĩa $\mu(x,y)=r_1(r_2(x)\circledast y)$ cho $x,y\in IoMT$, nhận thấy rằng $r_1(x)\circledast\mu(x,y)=r_1(xy)$. Theo đó, tính chất tách $split(r)=\langle r_1,\mu\rangle$ là một phép biến đổi stream $\text{BDLTrans}(IoMT,\text{FBDL}(IoMT))$.

Mô hình đại số của phép biến đổi BDL trong IoMT sử dụng một hàm đơn điệu, miền của nó là đơn điệu có lịch sử đầu vào hữu hạn. Một mô hình đại số như vậy có thể được dùng để biểu diễn sự biến đổi của một BDL vô hạn. Để lập luận cho quan điểm này, xét một hàm đơn điệu $\beta:IoMT_1^*\to IoMT_2^*$, $IoMT_1$ tương đương $IoMT_2$, đây là loại các frame đầu vào đồng thời cũng là các frame đầu ra tương ứng. Hàm β chỉ mở rộng cho hàm $\beta^\infty:IoMT\,{}_1^\infty\to IoMT\,{}_2^\infty$, ở đó $IoMT\,{}_1^\infty=IoMT_1^*\cup IoMT_1^1$ là

tập các frame hữu hạn và vô hạn trên $IoMT_1$, như vậy hàm $\beta^{\infty}(a_0a_1a_2a_3a_4a_5\dots)$ là tương đương với các frame theo quan hệ thứ tự $\beta(1) \preccurlyeq \beta(a_0) \preccurlyeq \beta(a_0a_1) \preccurlyeq \beta(a_0a_1a_2) \preccurlyeq \beta(a_0a_1a_2a_3) \preccurlyeq \beta(a_0a_1a_2a_3a_4) \preccurlyeq \cdots$.

2.6.7. Máy trạng thái của stream dạng BDL trong IoMT



Hình 2.14. Sơ đồ chuyển đổi trạng thái của một hệ thống livestream (IoMT)

Máy trạng thái stream dạng BDL trong IoMT được mô tả gồm có các thành phần (Hình 2.14) ở bảng 2.27 như sau:

Bảng 2.27. Mô tả máy trạng thái của stream dạng BDL

Thành phần	Mô tả
Mô hình tổng quát	Máy trạng thái xử lý stream được mô hình hóa bằng bộ 5-tuple $\mathcal{G} = (\mathrm{St}, \mathrm{init}, \mathrm{o}, \mathrm{next}, \mathrm{out})$.
St	Tập hợp các trạng thái của hệ thống (waiting, streaming, paused, ended).
init ∈ St	Trạng thái khởi đầu, nơi tính toán bắt đầu.
$o \in IoMT_2$	Hàm sinh frame/stream đầu ra đầu tiên.
$x \in IoMT_1$	Frame/stream đầu vào được tiêu thụ tuần tự theo thời gian.
$out(s, x) \in IoMT_2$	Phát sinh frame đầu ra tương ứng với trạng thái s và frame đầu vào x.
$next(s, x) \in St$	Chuyển đổi sang trạng thái kế tiếp sau khi xử lý frame x tại trạng thái s.
Ý nghĩa vận hành	Stream đầu vào và đầu ra thuộc tập IoMT ($IoMT_{1,2}$,⊛,1). Hệ thống thực hiện mô phỏng cơ chế hoạt động của livestream dạng BDL theo thời gian thực.
Ký hiệu trong sơ đồ trạng thái	- Trạng thái: Hình tròn. - Chuyển đổi trạng thái: Mũi tên. - Phép hợp stream (Phép toán stream hai ngôi): Ký hiệu ⊛.

Luận án mô hình hóa cơ chế hoạt động và tính toán stream dùng cho xử lý stream dạng BDL trong IoMT bằng các ký hiệu: init, out, s, o, x, next, ⊛ trên sơ đồ chuyển đổi trạng thái cho một hệ thống livestream (hình 2.14 và hình 2.15), với stream dạng BDL đầu vào và đầu ra là các frame của hai monoid sau: ($IoMT_1$,⊛,1) đầu vào và ($IoMT_2$,⊛,1) đầu ra.

Từ hai monoid này, luận án đề xuất một máy trạng thái dùng cho stream dạng BDL trong IoMT để tự động tạo ra các frame đầu ra, máy trạng thái có một không gian trạng thái vô hạn được ký hiệu là $\mathcal{G} = (\mathrm{St}, \mathrm{init}, \mathrm{o}, \mathrm{next}, \mathrm{out})$ và St là một tập hợp các trạng thái. Tính toán stream bắt đầu từ trạng thái đầu tiên tách biệt $\mathrm{init} \in \mathrm{St}$ và bắt đầu tạo ra một số stream dạng BDL đầu ra đầu tiên $\mathrm{o} \in IoMT_2$. Sau đó, phép tính stream tiếp tục tạo ra bằng cách tiêu thụ BDL đầu vào càng lúc càng tăng dần, tức là từng frame một. Một bước tính toán từ trạng thái $s \in \mathrm{St}$ bao gồm tiêu thụ frame đầu vào $x \in IoMT_1$ rồi phát sinh ra frame đầu ra tăng dần có dạng $\mathrm{out}(s, x) \in IoMT_2$ và biến đổi stream sang trạng thái tiếp theo $\mathrm{next}(s, x) \in \mathrm{St}$.

2.6.7.1. Máy trạng thái mô hình hóa stream dạng BDL trong IoMT

- **Mô tả máy trạng thái cụ thể và tổng quát:**
  - $\mathcal{G} = (\mathrm{St}, \mathrm{init}, \mathrm{o}, \mathrm{next}, \mathrm{out})$ dùng cho $(IoMT_1, \circledast, 1)$ đầu vào, $(IoMT_2, \circledast, 1)$ đầu ra.
  - $\mathrm{G}(IoMT_1, IoMT_2)$ đặc tả tập hợp các máy trạng thái xử lý stream BDL, ở đó có $\mathcal{G} = (\mathrm{St}, \mathrm{init}, \mathrm{o}, \mathrm{next}, \mathrm{out})$ (Hình 2.15).
  - *Trong đó:*
    - St là một tập hợp các trạng thái phi rỗng
    - $\mathrm{init} \in \mathrm{St}$ là trạng thái đầu tiên
    - $\mathrm{o} \in IoMT_2$ là đầu ra đầu tiên
    - $\mathrm{next}: \mathrm{St} \times IoMT_1 \to \mathrm{St}$ là mối quan hệ của phép biến đổi
    - $\mathrm{out}: \mathrm{St} \times IoMT_1 \to IoMT_2$ là mối quan hệ đầu ra.
    - Do vậy, một kiểu máy trạng thái $\mathrm{G}(IoMT_1, IoMT_2)$ được đặc tả để biểu diễn tập hợp máy trạng thái của stream đầu vào từ $IoMT_1$ và đầu ra từ $IoMT_2$.
- **Mô tả chuyển đổi trạng thái:**
  - Hàm chuyển đổi trạng thái tổng quát $\mathrm{gnext}: \mathrm{St} \times IoMT_1^* \to \mathrm{St}$ được mô tả:
    - $\mathrm{gnext}(s, 1) = s$ và
    - $\mathrm{gnext}(s, \langle x \rangle \circ \bar{y}) = \mathrm{gnext}(\mathrm{next}(s, x), \bar{y})$, trong đó $s \in \mathrm{St}$ có thể tiếp cận trong $\mathcal{G}$ nếu tồn tại BDL $\bar{x} \in IoMT_1^*$ sao cho $\mathrm{gnext}(\mathrm{init}, \bar{x}) = s$.
  - Mối quan hệ đầu ra tổng quát $\mathrm{gout}: \mathrm{St} \times IoMT_1^* \to IoMT_2$ được mô tả:
    - $\mathrm{gout}(s, 1) = 1$ và
    - $\mathrm{gout}(s, \langle x \rangle \circ \bar{y}) = \mathrm{out}(s, x) \circ \mathrm{gout}(\mathrm{next}(s, x), \bar{y})$.
  - Mối quan hệ đầu ra mở rộng $\mathrm{eout}: \mathrm{St} \times IoMT_1^* \to IoMT_2^*$ được mô tả:
    - $\mathrm{eout}(s, 1) = 1$ và
    - $\mathrm{eout}(s, \langle x \rangle \circ \bar{y}) = \langle \mathrm{out}(s, x) \rangle \circ \mathrm{eout}(\mathrm{next}(s, x), \bar{y})$.
- **Ví dụ 2.19:** Tính toán stream dạng BDL trên các monoid yêu cầu đầu ra tích lũy của $\mathcal{G}$ phải độc lập. Giả sử w là lịch sử đầu vào có thể phân chia thành hai tình huống: $w = u_1 \circ u_2 \circ u_3 \circ \cdots \circ u_m$ và $w = v_1 \circ v_2 \circ v_3 \circ \cdots \circ v_n$. Đầu ra của $\mathcal{G}$ có tính tích lũy khi tiêu thụ các frame $u_1, u_2, u_3, \ldots, u_m \approx v_1, v_2, v_3, \ldots, v_n$.

2.6.7.2. Máy trạng thái cho bộ đếm BDL trong IoMT

$(\text{FMultiset}(IoMT),\circledast,1)$ là một monoid đa tập hữu hạn đầu vào và $(\text{FBDL}(\mathbb{N}),\circledast,1)$ là một monoid BDL hữu hạn đầu ra dùng để đặc tả kiểu máy trạng thái $\text{G}(\text{FMultiset}(IoMT),\ \text{FBDL}(\mathbb{N}))$ (tiểu mục 2.6.6.1) với:

- $\text{St} = \mathbb{N}$: Tập hợp không gian trạng thái, vì máy trạng thái phải bảo toàn một bộ đếm để nhớ lại số lượng các frame đã được xem xét cho đến bây giờ
- $\text{init} = 0$ và $\text{o} = 1$: Trạng thái và đầu ra đầu tiên
- $\text{next}(s,\ x) = s + |x|,\ \forall s \in \text{St}$ và $\forall x \in \text{FMultiset}(IoMT)$: Biến đổi frame $|x|$ dạng BDL sang trạng thái s tiếp theo
- $\text{out}(s,\ \emptyset) = 1$: Đầu ra ban đầu rỗng theo trạng thái s
- $\text{out}(s,\ x) = \langle s+1, s+2, s+3, \ldots, s+|x| \rangle,\ x \neq \emptyset$: Biến đổi frame/stream x đầu ra sang trạng thái s tiếp theo.

2.6.7.3. Máy trạng thái cho bộ trộn BDL trong IoMT

$(\text{FMultiset}(IoMT),\circledast,1)$ kiểu đa tập hữu hạn với hai kênh đầu vào và một kênh một đầu ra dùng để đặc tả kiểu máy trạng thái $\text{G}(\text{FMultiset}(IoMT) \times \text{FMultiset}(IoMT),\ \text{FMultiset}(IoMT))$ (tiểu mục 2.6.6.2) với:

- $\text{St} = \text{Unit}$, với $\text{Unit} = \{\star\}$: Một tập singleton $\{x\}$ chứa một frame duy nhất
- $\text{init} = \ \star$: Trạng thái đầu tiên
- $\text{o} = \emptyset$: Đầu ra đầu tiên
- $\text{next}(s,\ x,\ y) = \star$: Mối quan hệ của phép biến đổi tăng bộ đếm chỉ có khả năng. Biến đổi frame/stream x dạng BDL sang trạng thái s tiếp theo.
- $\text{out}(s, \langle x,\ y \rangle) = x \cup y$ cho mọi bộ frame $x,\ y \in \text{FMultiset}(IoMT)$: Quan hệ đầu ra đặc tả sự lan truyền dẫn đến sự tăng trưởng đầu vào của cả kênh đầu vào và đầu ra.

2.6.7.4. Máy trạng thái cho bộ biến đổi phẳng BDL trong IoMT

$\text{Even}(IoMT) = (\text{St},\ \text{init},\ \text{o},\ \text{next},\ \text{out})$ là một máy trạng thái thực hiện biến đổi phẳng dùng để đặc tả kiểu máy trạng thái $\text{G}(\text{FBDL}(IoMT), IoMT)$ (tiểu mục 2.6.6.3) với:

- St = Unit và init=$\star$: Trạng thái đầu tiên rỗng. $\text{o} = 1_{IoMT}$: Đầu ra đầu tiên
- $\text{next}(s,\ x) = \ \star$: Mối quan hệ biến đổi frame x sang trạng thái s tiếp theo.
- $\text{out}(s,\ \langle a_1, a_2, a_3, \ldots, a_n \rangle) = a_1 \circ a_2 \circ a_3 \circ \cdots \circ a_n$: Mối quan hệ biến đổi frame a đầu ra sang trạng thái s tiếp theo.

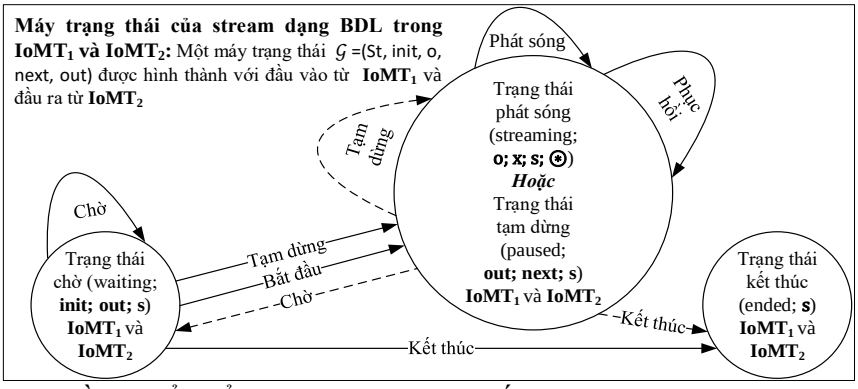
2.6.7.5. Máy trạng thái cho bộ tách BDL thành các lô trong IoMT

Cho $(IoMT,\circledast,1)$ là một monoid và một bộ tách $r = (r_1, r_2)$ các frame của BDL trong IoMT, đặc tả máy trạng thái $\text{Split}(r) = (\text{St},\ \text{init},\ \text{o},\ \text{next},\ \text{out})$ thực hiện biến đổi bộ tách $split(r)$: $\text{BDLTrans}(IoMT, \text{FBDL}(IoMT))$ (mục 2.6.6.4) với:

- $\text{St} = IoMT$: Máy trạng thái cần nhớ lại phần còn lại của đầu vào chưa đủ lô
- $\text{init} \ = 1_{IoMT}$: Trạng thái đầu tiên tách biệt

- o = 1: Đầu ra đầu tiên
- next$(s, x) = r_2(s \circ x)$: Mối quan hệ biến đổi cho bộ tách
- out$(s, x) = r_1(s \circ x)$: Mối quan hệ biến đổi đầu ra cho bộ tách

2.6.7.6. Mô phỏng và tương đương hai chiều của máy trạng thái BDL



Hình 2.15. Sơ đồ chuyển đổi trạng thái của hệ thống livestream (IoMT*1* và IoMT*2*)

Bảng 2.28. Mô tả sơ đồ chuyển đổi trạng thái của hệ thống livestream (Hình 2.15)

Thành phần	Mô tả
Mô hình tổng quát của máy trạng thái	Máy trạng thái xử lý stream BDL được mô hình hóa như sau: $\mathcal{G} =$ (St, init, o, next, out), với đầu vào thuộc $IoMT_1$ và đầu ra thuộc $IoMT_2$ (Hình 2.15).
Quan hệ mô phỏng hai chiều R	Quan hệ $R \subseteq$ St × St, thỏa: nếu $(s, t) \in R$ thì out$(s, x) =$ out(t, x) và (next(s, x), next$(t, x)) \in R$.
Tương đương hai chiều	Hai trạng thái $s \sim t$ nếu tồn tại một mô phỏng hai chiều R sao cho $s\ R\ t$.
Điều kiện tương đương	$\forall s, t \in$ St, $\forall x \in IoMT$, nếu $s \sim t$ thì đảm bảo: next$(s, x) \sim$ next(t, x).
Ý nghĩa trong IoMT	Hai trạng thái thuộc $IoMT_1$ và $IoMT_2$ được xem là tương đương hai chiều nếu chúng phản hồi đúng cùng một cách cho mọi frame x và cùng dẫn đến các trạng thái tương đương tiếp theo.
Hệ quả đối với livestream	- Cho phép thay thế trạng thái trong thiết kế hệ thống mà không ảnh hưởng hành vi livestream. - Tối ưu hóa pipeline mà vẫn giữ tính đúng đắn BDL. - Xác định rõ ràng các nhóm trạng thái tương đương để giảm độ phức tạp của máy trạng thái.
Ứng dụng	- Thiết kế máy trạng thái hiệu quả trong IoMT. - Xây dựng các bộ phận: streaming, paused, waiting, ended. - Kiểm chứng hoạt động livestream theo thời gian thực.
Ký hiệu trong sơ đồ trạng thái	- Trạng thái: Hình tròn. - Chuyển đổi trạng thái: Mũi tên liền. - Chuyển đổi ám chỉ có 02 hành động chuyển đổi tương ứng với 02 trạng thái tách biệt: Mũi tên đứt nét. - Phép hợp stream (Phép toán stream hai ngôi): Ký hiệu ⊛.

2.6.7.7. Tính nhất quán của các máy trạng thái cho BDL trong IoMT

Luận án xây dựng tập hợp các tính nhất quán mà một máy trạng thái phải tuân theo để đáp ứng các yêu cầu về frame BDL. Cho một máy trạng thái $\mathcal{G} = (\mathrm{St}, \mathrm{init}, \mathrm{o}, \mathrm{next}, \mathrm{out})$: $\mathrm{G}(IoMT_1, IoMT_2)$, luận án chỉ ra $\mathcal{G}$ phải có tính nhất quán nếu nó đáp ứng một số tính chất như sau:

- $\mathrm{next}(\mathrm{init}, 1) \sim \mathrm{init}$ (2.29)
- $\mathrm{next}(\mathrm{init}, xy) \sim \mathrm{next}(\mathrm{next}(\mathrm{init}, x), y)$, $\forall x, y \in IoMT_1$ (2.30)
- $\mathrm{o} \circ \mathrm{out}(\mathrm{init}, 1) = \mathrm{o}$ (2.31)
- $\mathrm{o} \circ \mathrm{out}(\mathrm{init}, xy) = \mathrm{o} \circ \mathrm{out}(\mathrm{init}, x) \circ \mathrm{out}(\mathrm{next}(\mathrm{init}, x), y)$, $\forall x, y \in IoMT_1$ (2.32)

$$\frac{\text{Các tính chất từ (2.29 - 2.32)}}{\text{Đảm bảo tính nhất quán của } \mathcal{G} = (\mathrm{St}, \mathrm{init}, \mathrm{o}, \mathrm{next}, \mathrm{out})}$$

Trong đó:

- $\mathcal{G}$ hoạt động không phụ thuộc vào cách đầu vào được chia thành các frame.
- Tính chất (2.30) chỉ ra phép biến đổi hai bước tiếp theo từ $\mathrm{init} \to^{x} s_1 \to^{y} s_2$ và phép biến đổi một bước tiếp theo từ $\mathrm{init} \to^{xy} t_1$ kết thúc ở các trạng thái (s_2 và t_1) sẽ dẫn đến hành vi tương tự trong tính toán stream của các dãy frame con tuần tự tiếp theo.
- Nếu không dù đầu vào xy được cung cấp cho máy trạng thái dưới dạng một frame xy đơn lẻ hoặc một BDL của hai frame x và y hoặc hai frame này được viết $\langle x, y \rangle$.

2.6.7.8. Yếu tố số hóa của máy trạng thái cho stream dạng BDL trong IoMT

Đặt $(IoMT, \circledast, 1)$ là một monoid, một yếu tố số hóa của một frame $x \in IoMT$ là một BDL có dạng $x_1, x_2, x_3, \ldots, x_n$ của các frame sao cho $x = x_1, x_2, x_3, \ldots, x_n$. BDL rỗng $\varepsilon \in IoMT^*$ là một yếu tố số hóa của 1, nếu không $x \in IoMT^*$ là một yếu tố số hóa của $x \in IoMT$ nếu $\pi(\bar{x}) = x$.

$$\frac{\mathrm{G}(IoMT_1, IoMT_2) \text{ được tạo bởi } \mathcal{G} = (\mathrm{St}, \mathrm{init}, \mathrm{o}, \mathrm{next}, \mathrm{out}), \forall x \in IoMT_1 \text{và } \forall \bar{x} \in IoMT_1^* \text{ của } x}{\mathcal{G} \text{ có tính nhất quán (2.33)}}$$

Trong đó: Máy trạng thái G được hình thành bởi máy trạng thái $\mathcal{G}$. Nếu $\mathcal{G}$ có tính nhất quán vào thời điểm $\forall x \in IoMT_1$ và tất cả các yếu tố số hóa $\bar{x} \in IoMT_1^*$ của x. Do đó, suy luận ra rằng $\mathcal{G}$ có tính nhất quán sau:

- $\mathrm{o} \circ \mathrm{gout}(\mathrm{init}, \bar{x}) = \mathrm{o} \circ \mathrm{out}(\mathrm{init}, x)$. (2.33)

Để lập luận cho tính chất (2.33), luận án đặc tả $\langle x_1, x_2, x_3, \ldots, x_n \rangle \in IoMT_1^*$ để đại diện cho một BDL hữu hạn các frame của $IoMT_1$. Các tính nhất quán được áp dụng cho tất cả $s \in \mathrm{St}$, $\bar{x} \in IoMT_1^*$ và $y \in IoMT_1$ sau:

- $\mathrm{gnext}(s, \bar{x} \circ \langle y \rangle) = \mathrm{next}(\mathrm{gnext}(s, \bar{x}), y)$ (2.34)
- $\mathrm{gout}(s, \bar{x} \circ \langle y \rangle) = \mathrm{gout}(s, \bar{x}) \circ \mathrm{out}(\mathrm{gnext}(s, \bar{x}), y)$ (2.35)
- $\mathrm{eout}(s, \bar{x} \circ \langle y \rangle) = \mathrm{eout}(s, \bar{x}) \circ \langle \mathrm{out}(\mathrm{gnext}(s, \bar{x}), y) \rangle$ (2.36)

Các tính chất từ (2.34) đến (2.36) có thể được lập luận bằng IH trên BDL $\bar{x}$. Xét một máy trạng thái BDL có tính nhất quán tùy chọn sau: $\mathcal{G} = (\mathrm{St}, \mathrm{init}, \mathrm{o}, \mathrm{next}, \mathrm{out})$ và $\mathcal{G}$ thỏa mãn tính nhất quán (2.37):

- $\text{gnext}(\text{init}, \langle x_1, \ldots, x_n \rangle) \sim \text{next}(\text{init}, x_1 \circ \cdots \circ x_n), \forall \langle x_1, \ldots, x_n \rangle \in IoMT_1^*$ (2.37)

Lập luận bằng IH về độ dài của BDL trong IoMT. Trường hợp cơ bản, đặc tả $\text{gnext}(\text{init}, 1) = \text{init}$ và $\text{next}(\text{init}, 1)$ mà nó tương đương hai chiều vì $\mathcal{G}$ có tính nhất quán xem tính chất (2.29). Với bước IH, suy luận ra các yếu tố số hóa sau:

- $\text{gnext}(\text{init}, \bar{x} \circ \langle y \rangle) = \text{next}(\text{gnext}(\text{init}, \bar{x}), y)$ [(2.34)] (2.38)
  $\sim \text{next}(\text{next}(\text{init}, \pi(\bar{x})), y)$ [Mở rộng và IH] (2.39)
  $\sim \text{next}(\text{init}, \pi(\bar{x}) \circ y),$ [(2.30)] (2.40)

Tính chất (2.40) tương đương với $\text{next}(\text{init}, \pi(\bar{x} \circ \langle y \rangle))$. Cuối cùng, điều này diễn giải cho tính chất (2.37), điều kiện này được lập luận bằng IH trên $\bar{x} \in IoMT_1^*$. Với trường hợp cơ bản, xem tính chất $\text{o} \circ \text{gout}(\text{init}, \varepsilon) = \text{o} \circ 1 = \text{o}$ là tương đương với $\text{o} \circ \text{out}(\text{init}, 1) = \text{o}$, xem tính chất (2.31) cho $\mathcal{G}$. Với bước IH, suy ra rằng:

- $\text{o} \circ \text{gout}(\text{init}, \bar{\text{x}} \circ \langle \text{y} \rangle) = \text{o} \circ \text{gout}(\text{init}, \bar{\text{x}}) \circ \text{out}(\text{gnext}(\text{init}, \bar{\text{x}}), \text{y})$ [(2.35)] (2.41)
  $= \text{o} \circ \text{out}(\text{init}, \pi(\bar{\text{x}})) \circ \text{out}(\text{gnext}(\text{init}, \bar{\text{x}}), \text{y})$ [IH] (2.42)
  $= \text{o} \circ \text{out}(\text{init}, \pi(\bar{\text{x}})) \circ \text{out}(\text{next}(\text{init}, \pi(\bar{\text{x}})), \text{y})$ [(2.37)] (2.43)
  $= \text{o} \circ \text{out}(\text{init}, \pi(\bar{\text{x}}) \circ \text{y}),$ [(2.32)] (2.44)

Trong đó tính chất (2.44) là tương đương với $\text{o} \circ \text{out}(\text{init}, \pi(\bar{x} \circ \langle y \rangle))$. Tính chất (2.33) khẳng định rằng có tính nhất quán đảm bảo tính đúng đắn cơ bản cho một máy trạng thái BDL, đầu ra mà chúng tạo ra là độc lập với phương pháp đặc tả mà đầu vào có thể được tách thành các frame.

Vậy $\mathcal{G}$ là một máy trạng thái, hàm $[\![G]\!]: IoMT_1^* \to IoMT_2^*$ đặc tả theo cách sau $[\![G]\!](\bar{x}) = \langle \text{o} \rangle \circ \text{eout}(\text{init}, \bar{x}), \forall \bar{x} \in IoMT_1^*$. Với $[\![\mathcal{G}]\!]$ là ký hiệu của $\mathcal{G}$. Đặc tả của $[\![\mathcal{G}]\!]$ là đề cập đến $[\![\mathcal{G}]\!](1) = \langle \text{o} \rangle$ và được áp dụng cho mọi $\bar{x} \in IoMT_1^*$ và $y \in IoMT_1$ qua tính chất (2.45) như sau:

- $[\![\mathcal{G}]\!](\bar{x} \circ \langle y \rangle) = [\![\mathcal{G}]\!](\bar{x}) \circ \langle \text{out}(\text{gnext}(\text{init}, \bar{x}), y) \rangle$ (2.45)

Khi đó $\mathcal{G}$ là có tính nhất quán, tính chất (2.33) cho biết rằng ký hiệu đề xuất cùng một đầu ra tích lũy với đầu vào của bất kỳ hai yếu tố số hóa nào. Vì vậy, các máy trạng thái $\mathcal{G}_1$ và $\mathcal{G}_2$ là tương đương nếu các ký hiệu của chúng tương đương và được hiểu là $[\![G_1]\!] = [\![G_2]\!]$.

2.6.7.9. Tính triển khai và tính nhất quán của máy trạng thái cho BDL

Cho $(IoMT_{1,2}, \circledast, 1)$ là hai monoid, $\mathcal{G}: \text{G}(IoMT_1, IoMT_2)$ là một máy trạng thái của BDL và $\langle \beta, \mu \rangle: \text{BDLTrans}(IoMT_1, IoMT_2)$ là hai hàm chuyển đổi BDL, cho thấy $\mathcal{G}$ triển khai $\langle \beta, \mu \rangle$, nếu $[\![\mathcal{G}]\!](\bar{x}) = \text{F}(\beta, \mu)(\bar{x}), \forall \bar{x} \in IoMT_1^*$.

$\mathcal{G}: \text{G}(IoMT_1, IoMT_2)$ có tính nhất quán nếu và chỉ nếu nó triển khai một số phép biến đổi stream dạng BDL. Để lập luận khẳng định này, giả sử cho $\mathcal{G} = (\text{St}, \text{init}, \text{o}, \text{next}, \text{out}): \text{G}(IoMT_1, IoMT_2)$ là một máy trạng thái có tính nhất quán, định nghĩa hàm $\beta: IoMT_1 \to IoMT_2$ dưới dạng $\beta(x) = \text{o} \circ \text{out}(\text{init}, x), \forall x \in IoMT_1$ và hàm $\mu: IoMT_1 \times IoMT_1 \to IoMT_2$ dưới dạng $\mu(x, y) = \text{out}(\text{next}(\text{init}, x), y)$,

$\forall x, y \in IoMT_1$, vậy có thể lập luận $\beta(x) \circ \mu(x, y) = \beta(xy)$. Nhận thấy, điều này đúng sau tính chất (2.32) phần nào của các thuộc tính có tính nhất quán cho $\mathcal{G}$. Do đó, $\langle \beta, \mu \rangle$ là một phép biến đổi BDL.

Phần còn lại cần lập luận $\mathcal{G}$ triển khai $\langle \beta, \mu \rangle$, được hiểu là $[\![\mathcal{G}]\!](\bar{x}) = \mathrm{F}(\beta, \mu)(\bar{x})$, $\forall \bar{x} \in IoMT_1^*$. Với trường hợp cơ bản, $[\![\mathcal{G}]\!](1) = \langle \mathrm{o} \rangle$ và $\mathrm{F}(\beta, \mu)(1) = \langle \beta(1) \rangle$, điều này giống nhau vì $\beta(1) = \mathrm{o} \circ \mathrm{out}(\mathrm{init}, 1) = \mathrm{o}$ theo tính chất (2.31). Với mỗi bước, ta tìm thấy các kết quả sau đây:

- $[\![\mathcal{G}]\!](\bar{x} \circ \langle y \rangle) = [\![\mathcal{G}]\!](\bar{x}) \circ \langle \mathrm{out}(\mathrm{gnext}(\mathrm{init}, \bar{x}), y) \rangle$ [(2.45)] (2.46)
- $\mathrm{F}(\beta, \mu)(\bar{x} \circ \langle y \rangle) = \mathrm{F}(\beta, \mu)(\bar{x}) \circ \langle \mu(\pi(\bar{x}), y) \rangle$ [Đặc tả của $\mathrm{F}(\beta, \mu)$] (2.47)

Theo IH đủ để lập luận $\mathrm{out}(\mathrm{gnext}(\mathrm{init}, \bar{x}), y)$ là tương đương với $\mu(\pi(\bar{x}), y) = \mathrm{out}(\mathrm{next}(\mathrm{init}, \pi(\bar{x})), y)$. Với $\mathrm{gnext}(\mathrm{init}, \bar{x})$ và $\mathrm{next}(\mathrm{init}, \pi(\bar{x}))$ là giống nhau theo tính chất (2.37). Ngược lại, giả sử $\mathcal{G} = (\mathrm{St}, \mathrm{init}, \mathrm{o}, \mathrm{next}, \mathrm{out})$: $\mathrm{G}(IoMT_1, IoMT_2)$ là một máy trạng thái triển khai $\langle \beta, \mu \rangle$: $\mathrm{BDLTrans}(IoMT_1, IoMT_2)$. Luận án mô tả mối quan hệ như sau: $R = \{(s, t) \in \mathrm{St} \times \mathrm{St} \mid \exists \bar{x}, \bar{y} \in IoMT_1^*$ với $\pi(\bar{x}) = \pi(\bar{y}), s = \mathrm{gnext}(\mathrm{init}, \bar{x})$, và $t = \mathrm{gnext}(\mathrm{init}, \bar{y})\}$. Do vậy, luận án định nghĩa rằng R là một mô phỏng hai chiều. Xét các trạng thái tùy ý của $s, t \in \mathrm{St}$ với sRt và $z \in IoMT$, tồn tại $\bar{x}, \bar{y} \in IoMT_1^*$ với $\pi(\bar{x}) = \pi(\bar{y})$ sao cho $s = \mathrm{gnext}(\mathrm{init}, \bar{x})$ và $t = \mathrm{gnext}(\mathrm{init}, \bar{y})$. Nhận thấy cần phải lập luận $\mathrm{out}(s, z) = \mathrm{out}(t, z)$ và $\mathrm{next}(s, z)\ R\ \mathrm{next}(t, z)$ là tương đương, nên lưu ý những tính chất sau:

- $[\![\mathcal{G}]\!](\bar{x} \circ \langle z \rangle) = [\![\mathcal{G}]\!](\bar{x}) \circ \langle \mathrm{out}(s, z) \rangle$ [(2.45), chỉ định s] (2.48)
- $\mathrm{F}(\beta, \mu)(\bar{x} \circ \langle z \rangle) = \mathrm{F}(\beta, \mu)(\bar{x}) \circ \langle \mu(\pi(\bar{x}), z) \rangle$ [Chỉ định của $\mathrm{F}(\beta, \mu)$] (2.49)

Bởi vì $\mathcal{G}$ triển khai $\langle \beta, \mu \rangle$ nên thu được $[\![\mathcal{G}]\!](\bar{x} \circ \langle z \rangle) = \mathrm{F}(\beta, \mu)(\bar{x} \circ \langle z \rangle)$ do đó $\mathrm{out}(s, z) = \mu(\pi(\bar{x}), z)$, tương tự như vậy có thể thu được $\mathrm{out}(t, z) = \mu(\pi(\bar{y}), z)$. Từ $\pi(\bar{x}) = \pi(\bar{y})$, suy ra $\mu(\pi(\bar{x}), z) = \mu(\pi(\bar{y}), z)$ như vậy $\mathrm{out}(s, z) = \mathrm{out}(t, z)$. Xét thấy rằng $s' = \mathrm{next}(s, z) = \mathrm{next}(\mathrm{gnext}(\mathrm{init}, \bar{x}), z) = \mathrm{gnext}(\bar{x} \circ \langle z \rangle)$ bằng việc sử dụng tính chất (2.34), tương tự như vậy có thể thu được $t' = \mathrm{next}(t, z) = \mathrm{gnext}(\bar{y} \circ \langle z \rangle)$. Từ đó, với $\pi(\bar{x} \circ \langle z \rangle) = \pi(\bar{x})z = \pi(\bar{y})z = \pi(\bar{y} \circ \langle z \rangle)$, suy ra $s'Rt'$ là mô phỏng hai chiều, suy ra R là một mô phỏng hai chiều.

Để lập luận $\mathcal{G}$ có tính nhất quán, chỉ cần đặc tả cho các trường hợp riêng ở tính chất (2.30) và (2.32). Đặt $x, y \in IoMT$, với tính chất (2.30) thì phải chỉ ra trạng thái $s = \mathrm{next}(\mathrm{next}(\mathrm{init}, x), y)$ và trạng thái $t = \mathrm{next}(\mathrm{init}, xy)$ là tương đương hai chiều. R là một tương quan mô phỏng hai chiều, lập luận $(s, t) \in R$ là đúng vì $s = \mathrm{gnext}(\mathrm{init}, \langle x, y \rangle), t = \mathrm{gnext}(\mathrm{init}, \langle xy \rangle)$ và $\pi(\langle x, y \rangle) = xy = \pi(\langle xy \rangle)$.

Từ tính chất (2.32), ta có $[\![\mathrm{G}]\!](\langle xy \rangle) = \langle \mathrm{o}, \mathrm{out}(\mathrm{init}, xy) \rangle$ và $\mathrm{F}(\beta, \mu)(\langle xy \rangle) = \langle \beta(1), \mu(1, xy) \rangle$ giống như $[\![\mathcal{G}]\!](\langle x, y \rangle) = \langle \mathrm{o}, \mathrm{out}(\mathrm{init}, x), \mathrm{out}(\mathrm{next}(\mathrm{init}, x), y) \rangle$ và $\mathrm{F}(\beta, \mu)(\langle x, y \rangle) = \langle \beta(1), \mu(1, x), \mu(x, y) \rangle$ bằng cách sử dụng các đặc tả của $[\![\mathrm{G}]\!]$ và F.

Vì $\mathcal{G}$ triển khai $\langle \beta, \mu \rangle$, nên nó được hiểu là $[\![\mathcal{G}]\!](\langle x, y \rangle) = \mathrm{F}(\beta, \mu)(\langle x, y \rangle)$ và $[\![\mathrm{G}]\!](\langle xy \rangle) = \mathrm{F}(\beta, \mu)(\langle xy \rangle)$. Do vậy chúng ta nhận được $\mathrm{o} \circ \mathrm{out}(\mathrm{init}, x) \circ \mathrm{out}(\mathrm{next}(\mathrm{init}, x), y) = \beta(1) \circ \mu(1, x) \circ \mu(x, y) = \beta(x) \circ \mu(x, y) = \beta(xy)$ và $\mathrm{o} \circ \mathrm{out}(\mathrm{init}, xy) = \beta(1) \circ \mu(1, xy) = \beta(xy)$. Như vậy, tính chất (2.32) có các tính nhất quán được giữ lại.

Tiểu mục 2.6.7.9 đã cung cấp một cách diễn giải cho các định nghĩa về các tính nhất quán cho các máy trạng thái của stream dạng BDL ở mục 2.6.7.7. Nó chỉ ra rằng các định nghĩa phù hợp tính đúng đắn, vì nó là các điều kiện cần, các điều kiện tính chính xác và các điều kiện tính đầy đủ cho một máy trạng thái của stream dạng BDL cho phép biến đổi BDL như hệ thống ngữ nghĩa các ký hiệu của chúng. Ngược lại, tính nhất quán cụ thể của một máy trạng thái có ngữ nghĩa mang tính biểu thị được định nghĩa rõ ràng về các phép biến đổi BDL trong IoMT.

2.6.7.10. Tính đầy đủ biểu cảm của các máy trạng thái cho BDL trong IoMT

Cho $\left(IoMT_{1,2}, \circledast, 1\right)$ là hai monoid và $\langle \beta, \mu \rangle$ là phép biến đổi stream $\mathrm{BDLTrans}(IoMT_1, IoMT_2)$, có tồn tại tính nhất quán của máy trạng thái cho việc triển khai $\langle \beta, \mu \rangle$ này. Để lập luận khẳng định này, xét tiểu mục 2.6.6 rằng hàm biến đổi μ MWF làm thỏa mãn tính chất sau: $\beta(x) \circ \mu(x, y) = \beta(xy)$, $\forall x, y \in IoMT$. Mô tả $\mathcal{G} = (\mathrm{St}, \mathrm{init}, \mathrm{o}, \mathrm{next}, \mathrm{out})$ gồm các thành phần: $\mathrm{St} = IoMT$, $\mathrm{init} = 1$, $\mathrm{o} = \beta(1)$, $\mathrm{next}(s, x) = s \circ x$ và $\mathrm{out}(s, x) = \mu(s, x)$ cho tất cả trạng thái $s \in \mathrm{St}$ và đầu vào $x \in IoMT$. Các tính chất sau đây đúng cho tất cả trạng thái $s \in \mathrm{St}$ và $\langle x_1, x_2, x_3, \ldots, x_n \rangle \in IoMT_1^*$.

- $\mathrm{gnext}(s, \langle x_1, x_2, x_3, \ldots, x_n \rangle) = s \circ x_1 x_2 x_3 \circ \cdots \circ x_n$ và (2.50)
- $\langle \mathrm{o} \rangle \circ \mathrm{eout}(\mathrm{init}, \langle x_1, x_2, x_3, \ldots, x_n \rangle) = \mathrm{F}(\beta, \mu)(\langle x_1, x_2, x_3, \ldots, x_n \rangle)$ (2.51)

Tính chất (2.50) và (2.51) được biểu diễn bằng IH trên dãy tuần tự frame có dạng $\langle x_1, x_2, x_3, \ldots, x_n \rangle$, suy ra $[\![G]\!](\bar{x}) = \langle \mathrm{o} \rangle \circ \mathrm{eout}(\mathrm{init}, \bar{x}) = \mathrm{F}(\beta, \mu)(\bar{x})$, $\forall \bar{x} \in IoMT_1^*$. Do đó, $\mathcal{G}$ triển khai được phép biến đổi $\langle \beta, \mu \rangle$, kết quả là $\mathcal{G}$ có một tính chất tự động nhất quán với tiểu mục 2.6.7.9. Tiểu mục 2.6.7.10 này khẳng định rằng, mô hình tính toán stream trừu tượng của máy trạng thái stream luôn có đầy đủ khả năng biểu cảm tính nhất quán để triển khai cho bất kỳ phép biến đổi stream nào.

2.6.7.11. Tính đúng đắn biến đổi phẳng của các máy trạng thái BDL

$\mathcal{G} = \mathrm{Even}(IoMT) = (\mathrm{Unit}, \star, 1_{IoMT}, \mathrm{next}, \mathrm{out})$ là máy trạng thái triển khai phép biến đổi $\langle \pi, \mu \rangle = even(IoMT)$ cho $(IoMT, \circledast, 1)$. Theo tiểu mục 2.6.6.3 và 2.6.7.4 lập luận bằng IH rằng $[\![\mathcal{G}]\!](\bar{x}) = \mathrm{F}(\pi, \mu)(\bar{x})$, $\forall \bar{x} \in \mathrm{FBDL}(IoMT)^*$. Ta có $[\![\mathcal{G}]\!](\varepsilon) = \langle 1_{IoMT} \rangle$ và $\mathrm{F}(\pi, \mu)(1) = \langle \pi(1) \rangle = \langle 1_{IoMT} \rangle$, $\forall \bar{x} \in \mathrm{FBDL}(IoMT)^*$ và $y \in \mathrm{FBDL}(IoMT)$. Do đó, $\mathrm{Even}(IoMT)$ được lập luận qua tính nhất quán (2.52) sau:

- $[\![\mathcal{G}]\!](\bar{x} \circ \langle y \rangle) = [\![\mathcal{G}]\!](\bar{x}) \circ \langle \mathrm{out}(\mathrm{gnext}(\mathrm{init}, \bar{x}), y) \rangle$ [Chỉ định của $[\![\mathcal{G}]\!]$] (2.52)

$= \mathrm{F}(\pi,\mu)(\bar{x}) \circ \langle \pi(y) \rangle$ [Chỉ định ra ngoài và IH] (2.53)
$= \mathrm{F}(\pi,\mu)(\bar{x}) \circ \langle \mu(\pi(\bar{x}), y) \rangle$ [Chỉ định của μ] (2.54)
$= \mathrm{F}(\pi,\mu)(\bar{x} \circ \langle y \rangle)$ [Chỉ định của F] (2.55)

2.6.7.12. Tính đúng đắn bộ tách của các máy trạng thái cho BDL

Cho máy trạng thái $\mathcal{G} = \mathrm{Split}(r) = (IoMT, 1_{IoMT}, \varepsilon, \mathrm{next}, \mathrm{out})$ triển khai phép biến đổi $\langle r_1, \mu \rangle = split(r)$ để tách $r = (r_1, r_2)$ cho $(IoMT, \circledast, 1)$ được thể hiện ở tiểu mục 2.6.6.4 và 2.6.7.5. Sử dụng tính chất của các bộ tách và IH để có $\mathrm{gnext}(\mathrm{init}, \bar{x}) = r_2(\pi(\bar{x}))$, $\forall \bar{x} \in IoMT_1^*$. Áp dụng IH để lập luận $[\![\mathcal{G}]\!](\bar{x}) = \mathrm{F}(r_1, \mu)(\bar{x})$ cho $\forall \bar{x} \in IoMT_1^*$. Ta có $[\![\mathcal{G}]\!](1) = \langle 1 \rangle$ và $\mathrm{F}(r_1, \mu)(1) = \langle r_1(1_{IoMT}) \rangle = \langle 1 \rangle$, cho $\forall \bar{x} \in IoMT_1^*$ và $y \in IoMT$. Như vậy, $\mathrm{Split}(r)$ có tính đúng đắn được lập luận qua tính nhất quán như sau:

- $[\![\mathcal{G}]\!](\bar{x} \circ \langle y \rangle) = [\![\mathcal{G}]\!](\bar{x}) \circ \langle \mathrm{out}(\mathrm{gnext}(\mathrm{init}, \bar{x}), y) \rangle$ [(2.45)] (2.56)
  $= \mathrm{F}(r_1, \mu)(\bar{x}) \circ \langle \mathrm{out}(r_2(\pi(\bar{x})), y) \rangle$ [IH] (2.57)
  $= \mathrm{F}(r_1, \mu)(\bar{x}) \circ \langle r_1(r_2(\pi(\bar{x})) \circ y) \rangle$ [Chỉ định của out] (2.58)
  $= \mathrm{F}(r_1, \mu)(\bar{x}) \circ \langle \mu(\pi(\bar{x}), y) \rangle$ [Chỉ định của μ] (2.59)
  $= \mathrm{F}(r_1, \mu)(\bar{x} \circ \langle y \rangle)$ [Chỉ định của F] (2.60)

2.6.8. Các bộ tổ hợp dùng cho máy trạng thái của BDL trong IoMT

- Luận án đề xuất 04 bộ tổ hợp cho các máy trạng thái stream dạng BDL trong IoMT để mô hình hóa cơ chế hoạt động của stream BDL như sau:
  - Nâng cao các hình thái tính toán stream dạng BDL.
  - Liên kết nối tiếp để biểu đạt song song hóa đường ống.
  - Liên kết song song để thể hiện nhiệm vụ song song.
  - Liên kết phản hồi để mô tả tính toán stream đầu ra phụ thuộc vào đầu ra trước đó.
- 04 bộ tổ hợp này được đặc tả cho phép biến đổi BDL và máy trạng thái BDL. Một số khía cạnh đại số trong IoMT được phân tích như sau:
  - Nâng cao hình thái triển khai qua máy trạng thái.
  - Liên kết song song và nối tiếp được triển khai qua tích trên các máy trạng thái. Liên kết song song thể hiện tính toán độc lập, trong khi liên kết nối tiếp chuyển đầu ra của liên kết đầu tiên làm đầu vào cho liên kết thứ hai.
  - Liên kết phản hồi diễn tiến qua vòng lặp được xác định rõ để tránh phân kỳ.

Luận án mô tả sự tương đương đúng đắn giữa các mức ngữ nghĩa và chương trình của các bộ tổ hợp trong tất cả các trường hợp sau: Nâng cao tính đồng cấu, liên kết song song hóa (Parallelism composition), liên kết nối tiếp, và liên kết phản hồi. Tính đúng đắn phù hợp để triển khai các liên kết như: *Elevate*, *Para*, *Serial*, và *Circle*. Do vậy, các frame được nhập vào là phù hợp với việc đặc tả mô đun hóa của các tính toán stream BDL, vì chúng có thể được hỗ trợ bởi các cấu trúc liên kết cần thiết để song song hóa và phân phối stream dạng BDL.

2.6.8.1. Nâng cao tính đồng cấu của các máy trạng thái cho BDL

Cho $h\colon (IoMT_1,\circledast,1) \to (IoMT_2,\circledast,1)$ là một đồng cấu monoid (HMonoid). Luận án tiến hành đặc tả Elevate(h) là một máy trạng thái có tính nhất quán và nó triển khai phép biến đổi elevate(h). Do đó, ta có tính chất (2.61) sau đây:

$$\frac{\text{HMonoid } h\colon (IoMT_1,\circledast,1) \to (IoMT_2,\circledast,1)}{\text{elevate}(h) = \langle \beta,\mu\rangle\colon \text{BDLTrans}(IoMT_1,IoMT_2)} \quad \begin{matrix}\beta(x) = h(x)\\ \mu(x,y) = h(y)\end{matrix}$$
$$\text{Elevate}(h) = (\text{St}, \text{init}, \text{o}, \text{next}, \text{out}), \tag{2.61}$$
Trong đó: $\text{St} = \text{Unit}$, $\text{init} = \star$, $\text{o} = h(1)$, $\text{next}(s,x) = s$, $\text{out}(s,x) = h(x)$.

2.6.8.2. Liên kết song song hóa của máy trạng thái cho BDL

Cho $\left(IoMT_{1,2,3,4},\circledast,1\right)$ là bốn monoid, $\langle \beta_1,\mu_1\rangle\colon \text{BDLTrans}(IoMT_1, IoMT_3)$ và $\langle \beta_2,\mu_2\rangle\colon \text{BDLTrans}(IoMT_2, IoMT_4)$ là phép biến đổi của stream và gọi $\mathcal{G}_1\colon \text{G}(IoMT_1, IoMT_3)$ và $\mathcal{G}_2\colon \text{G}(IoMT_2, IoMT_4)$ là hai máy trạng thái của stream (2.62). Do vậy, ta có hai tính chất sau đây:

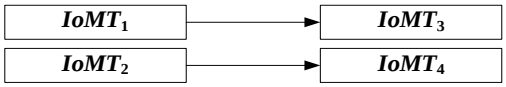
- *Triển khai:* Nếu $\mathcal{G}_1$ triển khai $\langle \beta_1,\mu_1\rangle$ và nếu $\mathcal{G}_2$ triển khai $\langle \beta_2,\mu_2\rangle$, thì Para$(\mathcal{G}_1,\mathcal{G}_2)$ triển khai $\langle \beta_1,\mu_1\rangle \parallel \langle \beta_2,\mu_2\rangle$.
- *Nhất quán:* Nếu $\mathcal{G}_1$ và $\mathcal{G}_2$ có tính nhất quán, thì Para$(\mathcal{G}_1,\mathcal{G}_2)$ cũng nhất quán.

$$\frac{\langle \beta_1,\mu_1\rangle\colon \text{BDLTrans}(IoMT_1, IoMT_3), \quad \langle \beta_2,\mu_2\rangle\colon \text{BDLTrans}(IoMT_2, IoMT_4)}{\langle \beta_1,\mu_1\rangle \parallel \langle \beta_2,\mu_2\rangle \;=\; \langle \beta,\mu\rangle\colon \text{BDLTrans}(IoMT_1 \times IoMT_2, IoMT_3 \times IoMT_4)}$$
- $\beta(x) = \beta_2\big(\beta_1(x)\big)$, $\mu(x,y) = \mu_2\big(\beta_1(x), \mu_1(x,y)\big)$,
- $\beta(\langle x_1,x_2\rangle) = \langle \beta_1(x_1), \beta_2(x_2)\rangle$ và $\mu(\langle x_1,x_2\rangle,\langle y_1,y_2\rangle) = \langle \mu_1(x_1,y_1), \mu_2(x_2,y_2)\rangle$

Máy trạng thái song song: $\text{Para}(\mathcal{G}_1,\mathcal{G}_2) = (\text{St}, \text{init}, \text{o}, \text{next}, \text{out})$ (2.62)

Trong đó:

- $\mathcal{G}_1 = (\text{St}_1, \text{init}_1, \text{o}_1, \text{next}_1, \text{out}_1)$, $\mathcal{G}_2 = (\text{St}_2, \text{init}_2, \text{o}_2, \text{next}_2, \text{out}_2)$,
- $\text{St} = \text{St}_1 \times \text{St}_2$, $\text{init} = \langle \text{init}_1, \text{init}_2\rangle$, $\text{o} = \langle \text{o}_1, \text{o}_2\rangle$,
- $\text{next}(\langle s_1,s_2\rangle,\langle a,c\rangle) = \langle \text{next}_1(\text{s}_1,\text{a}), \text{next}_2(s_2,c)\rangle$,
- $\text{out}(\langle s_1,s_2\rangle,\langle a,c\rangle) = \langle \text{out}_1(\text{s}_1,\text{a}), \text{out}_2(s_2,c)\rangle$
- Một thể hiện liên kết song song của bốn monoid $\left(IoMT_{1,2,3,4},\circledast,1\right)$ như sau (Hình 2.16):



Hình 2.16. Sơ đồ liên kết song song của bốn monoid

2.6.8.3. Liên kết nối tiếp của các máy trạng thái cho BDL trong IoMT

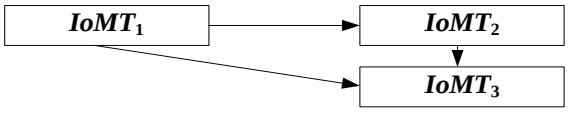
Cho $\left(IoMT_{1,2,3},\circledast,1\right)$ là ba monoid, $\langle \beta_1,\mu_1\rangle\colon \text{BDLTrans}(IoMT_1, IoMT_2)$ và $\langle \beta_2,\mu_2\rangle\colon \text{BDLTrans}(IoMT_2, IoMT_3)$ là hai phép biến đổi, và $\mathcal{G}_1\colon \text{G}(IoMT_1, IoMT_2)$ và $\mathcal{G}_2\colon \text{G}(IoMT_2, IoMT_3)$ là hai máy trạng thái (2.63). Do đó, ta có hai tính chất sau:

$$\frac{\langle\beta_1,\mu_1\rangle\colon \mathrm{BDLTrans}(\mathrm{IoMT}_1, \mathrm{IoMT}_2) \text{ và } \langle\beta_2,\mu_2\rangle : \mathrm{BDLTrans}(\mathrm{IoMT}_2, \mathrm{IoMT}_3)}{\langle\beta_1,\mu_1\rangle \gg \langle\beta_2,\mu_2\rangle = \langle\beta,\mu\rangle\colon \mathrm{BDLTrans}(\mathrm{IoMT}_1,\mathrm{IoMT}_3)}$$

$\mathrm{Serial}(\mathcal{G}_1, \mathcal{G}_2) = (\mathrm{St}, \mathrm{init}, \mathrm{o}, \mathrm{next}, \mathrm{out})$

Trong đó:

- $\mathcal{G}_1 = (\mathrm{St}_1, \mathrm{init}_1, \mathrm{o}_1, \mathrm{next}_1, \mathrm{out}_1)$, $\mathcal{G}_2 = (\mathrm{St}_2, \mathrm{init}_2, \mathrm{o}_2, \mathrm{next}_2, \mathrm{out}_2)$, $\mathrm{St} = \mathrm{St}_1 \times \mathrm{St}_2$,
- $\mathrm{init} = \langle \mathrm{init}_1, \mathrm{next}_2(\mathrm{init}_2, \mathrm{o}_1)\rangle$, $\mathrm{o} = \mathrm{o}_2 \circ \mathrm{out}_2\langle \mathrm{init}_1, \mathrm{o}_1\rangle$,
- $\mathrm{next}(\langle s_1, s_2\rangle, a) = \langle \mathrm{next}_1(s_1, \mathrm{a}), \mathrm{next}_2(s_2, \mathrm{out}_1(s_1, \mathrm{a}))\rangle$, $\mathrm{out}(\langle s_1, s_2\rangle, a) = \mathrm{out}_2(s_2, \mathrm{out}_1(s_1, c))$ (2.63)
- Một thể hiện mối liên kết nối tiếp của ba monoid $(IoMT_{1,2,3}, \circledast, 1)$ như sau (Hình 2.17):



Hình 2.17. Sơ đồ liên kết nối tiếp của ba monoid

- *Triển khai:* Nếu $\mathcal{G}_1$ triển khai $\langle\beta_1, \mu_1\rangle$ và $\mathcal{G}_2$ triển khai $\langle\beta_2, \mu_2\rangle$, thì $\mathrm{Serial}(\mathcal{G}_1, \mathcal{G}_2)$ triển khai $\langle\beta_1, \mu_1\rangle \gg \langle\beta_2, \mu_2\rangle$.
- *Nhất quán:* Nếu $\mathcal{G}_1$ và $\mathcal{G}_2$ có tính nhất quán, thì $\mathrm{Serial}(\mathcal{G}_1, \mathcal{G}_2)$ cũng nhất quán.

Để lập luận cho phát biểu trên, tính nhất quán được suy ra từ việc triển khai ở mục 2.6.7.9. Để lập luận việc triển khai, luận án xác định một số sự kiện. Đặc tả hàm $\mathrm{M}_2\colon IoMT_1^* \to IoMT_1$ như sau: $\mathrm{M}_2(1) = 1$, $\mathrm{M}_2(\langle x\rangle) = x$ cho $\forall x \in IoMT_1$, và $\mathrm{M}_2(\langle x, y\rangle \circ \bar{z}) = \langle xy\rangle \circ \bar{z}$ cho $\forall x, y \in IoMT_1$ và $\bar{z} \in IoMT_1^*$, $\mathcal{G}$ được đặc tả để đại diện cho liên kết nối tiếp của hai máy trạng thái $\mathcal{G}_1 \gg \mathcal{G}_2$.

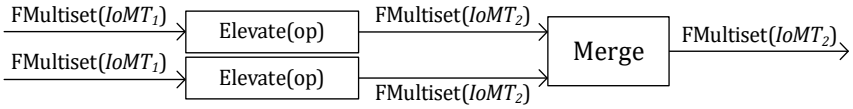
- $\mathrm{fst}(\mathrm{gnext}(s, \bar{x})) = \mathrm{gnext}_1(\mathrm{fst}(s), \bar{x})$, $\forall s \in \mathrm{St}$ và $\forall \bar{x} \in IoMT_1^*$ (2.64)
- $\mathrm{snd}(\mathrm{gnext}(s, \bar{x})) = \mathrm{gnext}_2(\mathrm{snd}(s), \mathrm{eout}_1(\mathrm{fst}(s), \bar{x}))$, $\forall s \in \mathrm{St}$ và $\forall \bar{x} \in IoMT_1^*$ (2.65)
- $[\![\mathcal{G}]\!](\bar{x}) = \mathrm{M}_2([\![\mathcal{G}_2]\!]([\![\mathcal{G}_1]\!](\bar{x})))$, $\forall \bar{x} \in IoMT_1^*$ (2.66)
- $\mathrm{F}(\beta, \mu)(\bar{x}) = \mathrm{M}_2(\mathrm{F}(\beta_2, \mu_2)(\mathrm{F}(\beta_1, \mu_1)(\bar{x})))$, $\forall \bar{x} \in IoMT_1^*$ (2.67)

Trong đó, với $\langle\beta, \mu\rangle = \langle\beta_1, \mu_1\rangle \gg \langle\beta_2, \mu_2\rangle$, bốn tính chất (2.64-2.67) có thể được lập luận bằng IH trên BDL $\bar{x}$. Tính chất (2.64) và (2.65) đòi hỏi lập luận tính chất (2.66). Luận án thiết lập máy trạng thái $\mathcal{G}$ triển khai $\langle\beta, \mu\rangle$ như sau:

- $[\![\mathcal{G}]\!](\bar{x}) = \mathrm{M}_2([\![\mathcal{G}_2]\!]([\![\mathcal{G}_1]\!](\bar{x})))$ [(2.66)] (2.68)
  $= \mathrm{M}_2([\![\mathcal{G}_2]\!](\mathrm{F}(\beta_1, \mu_1)(\bar{x})))$ [$\mathcal{G}_1$ triển khai $\langle\beta_1, \mu_1\rangle$] (2.69)
  $= \mathrm{M}_2(\mathrm{F}(\beta_2, \mu_2)(\mathrm{F}(\beta_1, \mu_1)(\bar{x})))$ [$\mathcal{G}_2$ triển khai $\langle\beta_2, \mu_2\rangle$] (2.70)
  $= \mathrm{F}(\beta, \mu)(\bar{x})$ [(2.67)] (2.71)

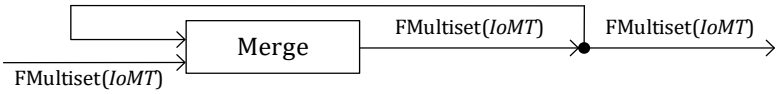
$\forall \bar{x} \in IoMT_1^*$ suy ra máy trạng thái $\mathcal{G}$ đã triển khai $\langle\beta, \mu\rangle$. Để lập luận cho nhận định này, luận án minh họa về cách xây dựng các phép tính stream phức hợp từ các phép tính stream cơ bản thông qua các bộ tổ hợp của các máy trạng thái BDL. Đặt $IoMT_1$, $IoMT_2$ là các tập hợp và $\mathrm{op}\colon IoMT_1 \to IoMT_2$ là một hàm. Phép tính stream BDL áp dụng op cho từng thành phần của $\mathrm{FMultiset}(IoMT_1)$, sinh ra hai đầu ra $\mathrm{FMultiset}(IoMT_2)$, sau đó trộn thành đầu ra $\mathrm{FMultiset}(IoMT_2)$. Hàm $\mathrm{op}\colon IoMT_1 \to IoMT_2$ được nâng thành đồng cấu monoid (HMonoid) như sau:

$\text{op}: \text{FMultiset}(IoMT_1) \to \text{FMultiset}(IoMT_2)$ với $\text{op}(x) = \{\text{op}(a) \mid a \in x\}$. Phép tính stream này được minh họa qua sơ đồ stream dạng BDL (Hình 2.18).



Hình 2.18. Sơ đồ thực hiện trộn (merge) hai monoid

Từ hình 2.18, đề xuất mỗi cạnh của sơ đồ có thể được biểu diễn bởi một kênh liên lạc mang BDL và được đánh dấu theo loại BDL tương ứng. Sơ đồ BDL này được đặc tả bằng $\mathcal{G} = \text{Serial}(\text{Para}(\text{Elevate}(\text{op}), \text{Elevate}(\text{op})), \text{Merge})$, trong đó Merge là máy trạng thái từ tiểu mục 2.6.7.3. Từ tiểu mục 2.6.8.1 đến 2.6.8.3, xác định $\mathcal{G}$ triển khai phép biến đổi $(\text{elevate}(\text{op}) \parallel \text{elevate}(\text{op})) \gg \text{merge}$, với merge là phép trộn đa kênh đầu vào cho BDL được trình bày ở tiểu mục 2.6.6.2.



Hình 2.19. Sơ đồ chu trình phản hồi stream trong IoMT

Kiểm tra máy trạng thái tổ hợp phản hồi tạo ra các chu kỳ của sơ đồ stream $\text{Merge}: \text{G}(\text{FMultiset}(IoMT) \times \text{FMultiset}(IoMT), \text{FMultiset}(IoMT))$ được trình bày ở tiểu mục 2.6.7.3. Sơ đồ (Hình 2.19) minh họa trực quan chu trình phản hồi stream trong IoMT, nơi kênh đầu ra của Merge được nối lại với một kênh đầu vào của nó, tạo thành chu trình phản hồi.

Sơ đồ (Hình 2.19) với đầu vào là một frame đơn {a} trên kênh đầu vào thứ nhất của máy trạng thái Merge sẽ khiến Merge phát ra frame {a}, sau đó được gửi lại vào kênh đầu vào thứ hai. Quá trình này tạo thành vòng lặp vô tận của việc tiêu thụ và phát ra frame {a}, dẫn đến hiện tượng phân kỳ hay còn gọi là vòng lặp vô hạn. Hành vi này không phù hợp với các hệ thống xử lý stream, vì phân kỳ có thể gây hệ thống không phản hồi.

Luận án đề xuất cơ chế phản hồi loại bỏ để giải quyết vòng lặp vô hạn trong stream. Phép tính stream được thực hiện tuần tự qua nhiều vòng lặp, đảm bảo tiến hành dựa trên dữ liệu đầu vào.

Giải pháp là cung cấp bộ tách lô (tiểu mục 2.6.6.4) chia BDL đầu vào thành các lô, mỗi vòng lặp phản hồi xử lý một lô dữ liệu, tạo lô đầu ra và gửi vào vòng lặp tiếp theo. Cách tiếp cận này giúp tổ chức mỗi lô dữ liệu tương ứng một vòng lặp.

2.6.8.4. Liên kết phản hồi của máy trạng thái cho BDL trong IoMT

Cho $\left(IoMT_{1,2}, \circledast, 1\right)$ là hai cấu trúc monoid, một phép biến đổi stream $\langle \beta, \mu \rangle: \text{BDLTrans}(IoMT_1, IoMT_2)$, máy trạng thái $\mathcal{G}: \text{G}(IoMT_1, IoMT_2)$ và $r = (r_1, r_2)$ là bộ tách BDL thành các lô trong $IoMT_1$ và $IoMT_2$ xem ở tiểu mục 2.6.6.4. Do đó, luận án có hai tính chất cho liên kết phản hồi sau đây:

- *Triển khai:* Nếu $\mathcal{G}$ triển khai $\langle\beta,\mu\rangle$, thì $\mathrm{Circle}(\mathcal{G},r)$ triển khai $circle(\beta,\mu,r)$.
- *Nhất quán:* Nếu $\mathcal{G}$ có tính chất nhất quán, thì $\mathrm{Circle}(\mathcal{G},r)$ cũng nhất quán.

Để lập luận cho hai tính chất trên, với liên kết nối tiếp đủ để lập luận máy trạng thái $\mathrm{Circle}(\mathcal{G})$ triển khai $\langle\gamma,\nu\rangle = circle(\beta,\mu)$ bằng tính chất triển khai. Với $\mathrm{G}(\mathrm{FBDL}(IoMT_1),\mathrm{FBDL}(IoMT_2))$, đủ để đặc tả các hàm biến đổi và đầu ra trên các frame dạng BDL đơn lẻ như trong các phát biểu từ (2.61) đến (2.72).

$$\frac{\langle\beta,\mu\rangle\colon \mathrm{BDLTrans}(IoMT_1\times IoMT_2, IoMT_2)}{\begin{aligned}
circle(\beta,\mu) &= \langle\gamma,\nu\rangle\colon \mathrm{BDLTrans}(\mathrm{FBDL}(IoMT_1),\mathrm{FBDL}(IoMT_2))\\
\gamma(\langle a_1,\dots,a_n\rangle) &= \langle b_0,b_1,\dots,b_n\rangle\\
\gamma(1) &= \langle b_0\rangle\\
\textit{Trong đó: } b_0 &= \beta(1_{IoMT_1},1_{IoMT_2})\\
\gamma(\langle a_1,\dots,a_n,a_{n+1}\rangle) &= \gamma(\langle a_1,\dots,a_n\rangle)\circ\langle b_{n+1}\rangle\\
\textit{Trong đó: } b_{n+1} &= \mu(\langle a_1\circ\cdots\circ a_n, b_0b_1\circ\cdots\circ b_{n-1}\rangle,\langle a_{n+1},b_n\rangle)
\end{aligned}}$$

Máy trạng thái $\mathrm{Circle}(\mathcal{G}) = (\mathrm{St}',\mathrm{init}',\mathrm{o}',\mathrm{next}',\mathrm{out}')\colon \mathrm{G}(\mathrm{FBDL}(IoMT_1),\mathrm{FBDL}(IoMT_2))$

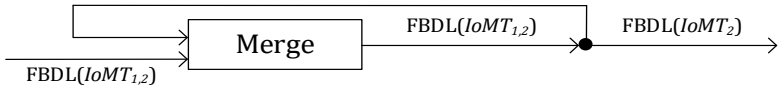
Trong đó: (2.72)

- $\mathcal{G} = (\mathrm{St},\mathrm{init},\mathrm{o},\mathrm{next},\mathrm{out})\colon \mathrm{G}(IoMT_1\times IoMT_2, IoMT_2)$
- $\mathrm{St}' = \mathrm{St}\times IoMT_2$, liên kết thứ hai: lô đầu ra cuối cùng, $\mathrm{init}' = \langle\mathrm{init},\mathrm{o}\rangle$,
- $\mathrm{o}' = \langle\mathrm{o}\rangle$, $\mathrm{next}'(\langle s,b\rangle,a) = \langle\mathrm{next}(s,\langle a,b\rangle),\mathrm{out}(s,\langle a,b\rangle)\rangle$,
- $\mathrm{out}'(\langle s,b\rangle,a) = \langle\mathrm{out}(s,\langle a,b\rangle)\rangle$

$$\frac{\langle\beta,\mu\rangle\colon \mathrm{BDLTrans}(IoMT_1\times IoMT_2, IoMT_2) \text{ và bộ tách } r \text{ cho } IoMT_1}{circle(\beta,\mu,r) = split(r)\gg circle(\beta,\mu)\gg even(IoMT_2)\colon \mathrm{BDLTrans}(IoMT_1, IoMT_2)}$$

$$\frac{\mathcal{G}\colon \mathrm{G}(IoMT_1\times IoMT_2, IoMT_2) \text{ và bộ tách } r \text{ cho } IoMT_1}{\mathrm{Circle}(\mathcal{G},r) = \mathrm{Serial}(\mathrm{Split}(r),\mathrm{Circle}(\mathcal{G}),\mathrm{Even}(IoMT_2))\colon \mathrm{G}(IoMT_1, IoMT_2)}$$

- Liên kết phản hồi của máy trạng thái Circle cho BDL trong hai monoid $(IoMT_{1,2},\circledast,1)$ bằng máy trạng thái liên kết nối tiếp Serial (gồm máy trạng thái tách, máy trạng thái phản hồi, và máy trạng thái phẳng) như sau (Hình 2.20):



Hình 2.20. Sơ đồ liên kết phản hồi của máy trạng thái Circle cho BDL

Ví dụ 2.20: Minh họa sử dụng tổ hợp liên kết phản hồi (2.72), xét hai hàm $\langle\beta,\mu\rangle$ để thêm hai BDL đầu vào hữu hạn. Hàm $\beta\colon \mathrm{FBDL}(IoMT)\times\mathrm{FBDL}(IoMT)\to\mathrm{FBDL}(IoMT)$ với hai kênh đầu vào hữu hạn và một kênh đầu ra được định nghĩa: $\beta(x_1\circ x_2\circ\cdots\circ x_m, y_1\circ y_2\circ\cdots\circ y_n) = (x_1+y_1)\circ(x_2+y_2)\circ\cdots\circ(x_k+y_k)$, với $k=\min(m,n)$. Xét bộ tách $r = (r_1,r_2)$, ở đó mỗi lô là một frame đơn: $r_1(x_1\circ\cdots\circ x_n) = \langle x_1,\dots,x_n\rangle$ và $r_2(x_1\circ\cdots\circ x_n) = \varepsilon$, bộ tách này giúp mỗi chu trình tính toán tiêu thụ từng frame dạng BDL. Do vậy, phép biến đổi $circle(\beta,\mu,r) = \langle\gamma,\nu\rangle$ biểu diễn trộn liên tục bằng hàm $\gamma(x_1\circ\cdots\circ x_n) = x_1\circ(x_1+x_2)\circ\cdots\circ(x_1+\cdots+x_n)$.

2.7. Kết luận chương 2

Chương 2 này đã xây dựng một số khía cạnh đại số bao gồm: xây dựng mười phép toán xử lý stream, tổ hợp các phép toán này thành các biểu thức stream, mở rộng đại số stream và đồng đại số stream, áp dụng mười phép toán xử lý stream này vào xây dựng cấu trúc đại số monoid và tính chất monoid dùng cho stream dạng BDL trong IoMT. Cấu trúc đại số monoid này sẽ hình thức hóa cơ chế hoạt động của stream bằng cách áp dụng mười phép toán xử lý stream đã được luận án xây dựng ở *tiểu mục 2.3.3* và *tiểu mục 2.3.4* thông qua tính chất kết hợp và tính chất đồng nhất stream. Tính chất monoid hình thức hóa cơ chế hoạt động của stream bằng các phép biến đổi stream, máy trạng thái stream, và các bộ tổ hợp máy trạng thái stream.

Chương 3 tập trung trình bày cơ chế lập lịch để xử lý stream dạng BDL trong IoMT. Việc lập lịch này cho phép cơ chế đường ống (pipeline) hoạt động mà ở đó tất cả các tài nguyên phần cứng bao gồm các thanh ghi cùng với các đơn vị chức năng được tái sử dụng trong suốt các bước điều khiển khác nhau giúp tối ưu hóa tài nguyên phần cứng. Tính toán dựa trên phép tính stream dạng BDL trong IoMT được xem như một hàm từ $IoMT^{\omega} \rightarrow IoMT^{\omega}$ trong đại số stream, nó hỗ trợ việc biểu đồ hóa mạng lưới RTL cho stream dạng BDL trong IoMT.

Chương 3. ĐẶC TẢ HÌNH THỨC VÀ TIẾP CẬN ĐỒNG QUY NẠP ĐỂ KIỂM CHỨNG TỔNG HỢP TÍNH TOÁN BDL DỰA TRÊN PHÉP TÍNH STREAM

Chương này tập trung trình bày đóng góp tạo ra cơ chế lập lịch để xử lý stream dạng BDL qua việc áp dụng mười phép toán xử lý stream đã được công bố trong công trình [CT.2] của luận án. Cơ chế lập lịch này cần có 04 bước sau: Phân hoạch các phép toán stream; Lập lịch cho tính toán BDL; Cấp phát và liên kết thanh ghi; Cấp phát và liên kết đơn vị chức năng. Các bước này cho phép cơ chế đường ống hoạt động mà ở đó tất cả các tài nguyên phần cứng bao gồm các thanh ghi cùng với các đơn vị chức năng được tái sử dụng trong suốt các bước điều khiển khác nhau nhằm giảm độ trễ, tăng thông lượng, cân bằng tải và tối ưu sử dụng tài nguyên hệ thống.

3.1. Giới thiệu

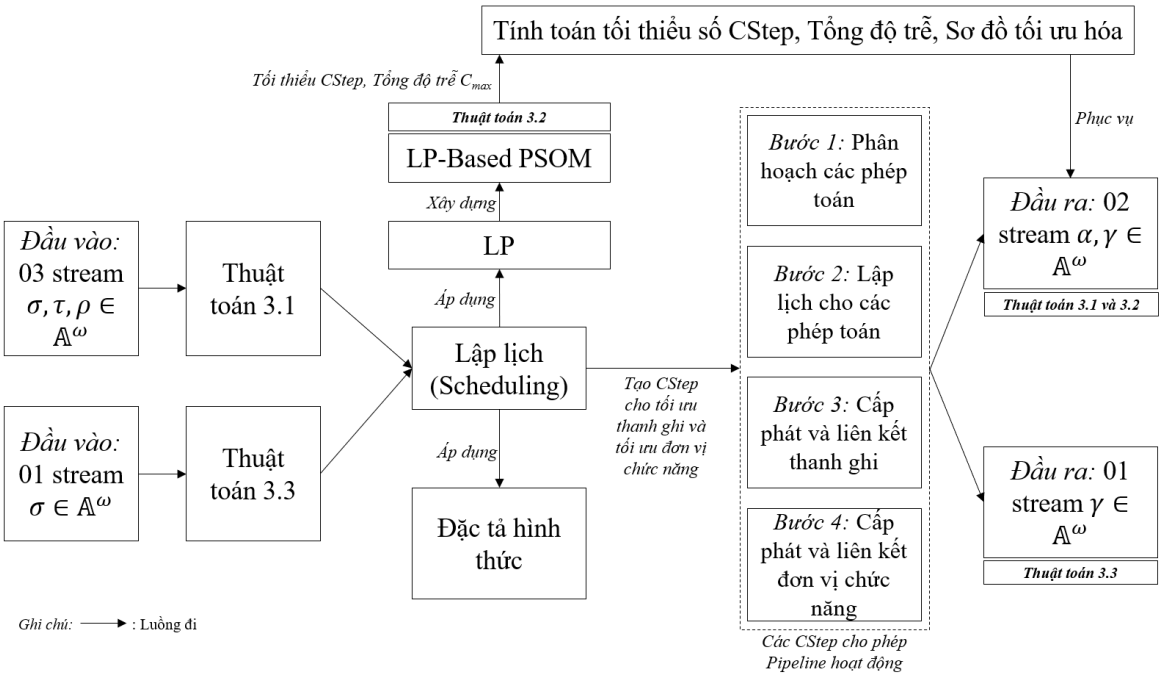
Phân tích stream dạng BDL là quá trình xử lý dữ liệu lớn phát sinh liên tục theo thời gian thực từ các công nghệ như IoT và IoMT [CT.5][CT.7][82], đòi hỏi kiến thức chuyên sâu và công cụ hỗ trợ phù hợp. Các vấn đề như phân tích, lập lịch stream dạng BDL và kiểm chứng hình thức stream dạng BDL vẫn chưa được giải quyết triệt để. Đặc biệt, khám phá những quy luật và cơ chế hoạt động của stream dạng BDL bằng các công nghệ hiện có hoặc xây dựng nền tảng lý thuyết nhằm giải thích cơ chế hoạt động là nhiệm vụ đầy thách thức. Sự tích hợp giữa công nghệ xử lý theo lô và xử lý stream phát trực tuyến còn hạn chế, ảnh hưởng đến hiệu quả tổng thể. Do đó, cần phát triển nền tảng xử lý dữ liệu lớn để đáp ứng yêu cầu việc lập lịch và kiểm chứng stream dạng BDL phục vụ các mục tiêu chiến lược lâu dài [83].

Chương 3 này tập trung vào nghiên cứu cơ chế lập lịch để xử lý stream dạng BDL trong IoMT. Các bước trong lập lịch sẽ cho phép cơ chế đường ống hoạt động mà ở đó tất cả các tài nguyên bao gồm các thanh ghi cùng với các đơn vị chức năng được tái sử dụng trong suốt các bước điều khiển khác nhau nhằm giảm độ trễ, tăng thông lượng, cân bằng tải và tối ưu sử dụng tài nguyên hệ thống. Tính toán dựa trên phép tính stream dạng BDL trong IoMT là một hàm từ $IoMT^{\omega}$ đến $IoMT^{\omega}$ trong đại số stream, nó hỗ trợ việc biểu đồ hóa mạng lưới RTL cho stream dạng BDL. Một mạng lưới như vậy có thể được thiết kế như các triển khai của tính toán stream.

3.2. Đặc tả hình thức tính toán BDL

Biểu diễn tính toán stream dạng BDL dựa trên phép toán stream cho phép đặc tả khái niệm đường ống hiệu quả ở đó tất cả các tài nguyên phần cứng bao gồm các thanh ghi và các đơn vị chức năng (FUs) được tái sử dụng lại trong suốt các bước (CSteps) tính toán khác nhau (Hình 3.1). Trong phép tính stream, tính toán BDL dựa trên phép tính stream được xem như là hàm từ tập các stream $\mathbb{A}^{\omega}$ tới tập các stream

$\mathbb{A}^{\omega}$ giúp cho việc biểu diễn cho các mạng RTL trực quan. Các mạng như vậy có thể được triển khai cho tính toán BDL dựa trên phép tính stream.



Hình 3.1. Sơ đồ đặc tả hình thức tính toán stream dạng BDL

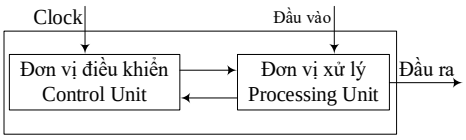
Hình 3.1 là sơ đồ đặc tả hình thức tính toán BDL của thuật toán 3.1 và 3.3. Thuật toán 3.1 với 3 stream đầu vào, tiến hành áp dụng đặc tả hình thức để lập lịch qua bốn bước tạo các bước điều khiển (CStep) cho phép pipeline hoạt động mà ở đó tất cả các tài nguyên phần cứng, bao gồm: Các thanh ghi và các FUs được tái sử dụng trong suốt các CStep khác nhau *(Thực hiện ở tiểu mục 3.2.2)*. Ngoài ra, luận án còn tiến hành lập lịch bằng cách xây dựng mô hình tối ưu hóa lập lịch theo cơ chế đường ống dựa trên quy hoạch tuyến tính (LP-Based PSOM), mô hình này được tạo ra từ các ràng buộc phụ thuộc và các ràng buộc không chồng lắp *(LP-Based PSOM được hiện thực bằng Thuật toán 3.2)* giúp tính toán tối thiểu số CStep, tổng độ trễ và cho ra sơ đồ tối ưu hóa lập lịch *(Thực hiện ở tiểu mục 3.2.2.5)*.

3.2.1. Giới thiệu ngôn ngữ RTL

Dựa vào *Tiểu mục 1.6.10* của *Chương 1*, ngôn ngữ RTL [46] là một biểu diễn trung gian rất gần với ngôn ngữ lập trình hợp ngữ, giống như ngôn ngữ được sử dụng trong một trình biên dịch. Nó được sử dụng để mô tả stream tại RTL của một kiến trúc. RTL là một sự trừu tượng được sử dụng để xác định các thành phần số học của một thiết kế và là sự trừu tượng chính được sử dụng để xác định các hệ thống điện tử tự động hiện nay và thường được sử dụng như mô hình mẫu trong việc thiết kế và việc kiểm chứng tính toán BDL dựa trên phép tính stream.

Mọi hệ thống được mô hình hóa bằng cách sử dụng một FU nhận thông tin từ môi trường bên ngoài thông qua các tín hiệu và xử lý chúng để tạo ra một tín hiệu

đầu ra, được giải thích như là phản ứng đối với môi trường. Mỗi đơn vị chức năng được triển khai bởi một đơn vị điều khiển và đơn vị xử lý. Đơn vị đầu tiên cung cấp một tín hiệu để đồng bộ hóa các phép toán được thực hiện bởi đơn vị thứ hai. Toàn bộ hệ thống dựa trên một đồng hồ duy nhất, cung cấp đồng bộ. Giả định cơ bản là mạch phải ổn định trước khi chu kỳ đồng hồ kết thúc. Hình 3.2 cho thấy mô đun RTL của một FU trong hệ thống số thông thường, được xác định như sau:



Hình 3.2. Mô dun RTL của một FU trong hệ thống kỹ thuật số

- Các thành phần chứa các câu lệnh của các phần tạo nên đơn vị xử lý.
- Dãy tuần tự điều khiển xác định trình tự các lệnh nội bộ cần phải được phát ra bởi đơn vị điều khiển.
- Phân công cố định xác định một phép toán phải được lặp lại mỗi chu kỳ đồng hồ.

Dãy tuần tự điều khiển được tạo thành từ các bước. Mỗi bước được đánh số và phải được thực hiện trong một đơn vị đồng hồ. Mỗi bước được tạo thành từ một hoặc nhiều lệnh được thực hiện song song. Tất cả các lệnh trong một bước được tách bằng dấu chấm phẩy, ký hiệu ";". Do đó, dãy tuần tự điều khiển có dạng sau:

$i: op_1; op_2; op_3$
$j: op_4; op_5$

Trong đó i và j là bước i và bước j chung, còn op_i là các lệnh riêng biệt. Các cấu trúc chính của ngôn ngữ là điều kiện và gán giá trị. Phép gán mô tả việc truyền giá trị giữa hai thanh ghi. Phía bên phải của phép gán có thể chứa bất kỳ toán tử logic nào và các phép toán. Các phép toán trên được mô tả như sau.

$i: targetRegister := sourceRegister$
$j: targetRegister := sourceRegister_1$ và $sourceRegister_2$
$k: if(c_1;\ c_2)\ then\ (op_1; op_2)$
$h: if(c_3;\ c_4)\ goto\ (n; m)$

Để biểu diễn một kết nối trực tiếp giữa các phần tử, ngôn ngữ này cho phép ta biểu diễn phép toán gán giá trị cho một lệnh. Trong trường hợp này, phép toán gán chỉ có hiệu lực trong một chu kỳ đồng hồ. Nó được biểu diễn bởi phép toán gán được ký hiệu là ": = " và được sử dụng để biểu diễn phép gán cho các lệnh đầu ra.

3.2.2. Mức chuyển đổi thanh ghi với thuật toán 3.1

Trong ngữ cảnh mức chuyển đổi thanh ghi (RTL), thuật toán 3.1 (phi vòng lặp) được sử dụng để khảo sát các mức chuyển đổi thanh ghi. Các bước tổng hợp khác nhau của việc tính toán BDL dựa trên phép tính stream được minh họa qua một bộ 03 stream đầu vào ($\sigma, \tau, \rho\ \in \mathbb{A}^\omega$) thành 02 stream đầu ra ($\alpha, \gamma \in \mathbb{A}^\omega$), như được định nghĩa bằng thuật toán 3.1 như sau [CT.2]:

Thuật toán 3.1: Xử lý tính toán 03 stream đầu vào bằng các phép toán xử lý stream *(phi vòng lặp)* kết quả đầu ra là 02 stream.

Đầu vào : $\sigma, \tau, \rho \in \mathbb{A}^{\omega}$

Đầu ra : $\alpha, \gamma \in \mathbb{A}^{\omega}$

1. Begin

2. $\alpha := \Big(zip(\sigma + \tau) + take\big(\rho + (\tau + \rho)\big) + \big(drop(\sigma + \tau) + \tau\big)\Big);$

3. $\gamma := \Big(split(\sigma) + zip(\sigma + \tau) + \Big(rev(\sigma + \tau) + take\big(\rho + (\tau + \rho)\big)\Big)\Big);$

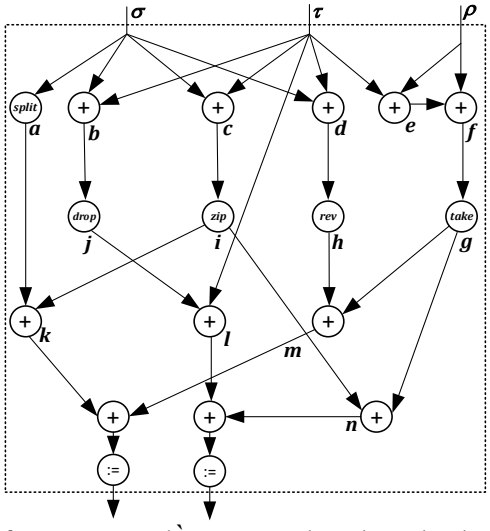
4. End

Thuật toán 3.1, cho ba stream đầu vào (số lượng stream có thể tùy ý) thuộc tập các stream $\mathbb{A}^{\omega}$, thuật toán 3.1 được phân hoạch bằng các phép toán một ngôi và các phép toán hai ngôi *(Tiểu mục 2.3.1, Chương 2)* tùy ý trong biểu diễn các stream đầu vào và các stream đầu ra dạng BDL trong IoMT. Trong lập lịch theo cơ chế đường ống (pipeline): Phân hoạch các phép toán xử lý stream thành các nhóm khối xử lý riêng trong biểu thức đại số stream, các khối không chồng lấn, toàn bộ các khối hợp lại tạo thành biểu thực đại số stream mới.

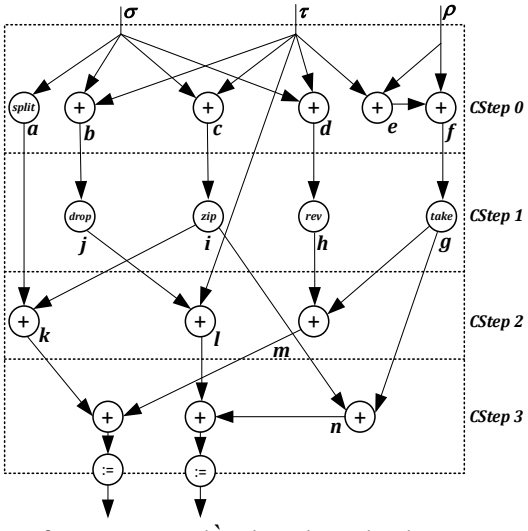
Luận án đề xuất cơ chế lập lịch gồm 04 bước sau: Phân hoạch các phép toán stream, Lập lịch cho tính toán BDL, Cấp phát và liên kết thanh ghi, Cấp phát và liên kết đơn vị chức năng. Các bước này cho phép cơ chế đường ống hoạt động mà ở đó tất cả các tài nguyên phần cứng bao gồm các thanh ghi cùng với các đơn vị chức năng được tái sử dụng trong suốt các bước điều khiển khác nhau giúp tối ưu hóa tài nguyên.

3.2.2.1. Phân hoạch các phép toán stream dạng BDL

Phân hoạch các phép toán xử lý stream vào sơ đồ stream dạng BDL tương ứng với mã code của thuật toán 3.1 để tạo ra hình 3.3, ở đó các kết quả trung gian được đặt tên một cách trực tiếp lên sơ đồ stream gồm: $a, b, c, d, e, f, g, h, i, j, k, l, m$ và n.



Hình 3.3. Sơ đồ stream phân hoạch phép toán stream cho tính toán BDL



Hình 3.4. Sơ đồ phân hoạch phép toán stream cho tính toán BDL thành các CStep

3.2.2.2. Lập lịch cho tính toán BDL

Việc lập lịch cho tính toán BDL dựa trên phép tính stream là một tiến trình đặc tả một số bước tính toán $k \geq 0$ được yêu cầu cho việc tính toán ứng dụng và phân phối từng phép toán stream của ứng dụng vào từng CStep cụ thể và được đánh số cho mỗi CStep là $0, 1, 2, 3, \ldots k$. Trong hình 3.4, việc lập lịch cho tính toán BDL dựa trên phép tính stream được xác định là bốn bước CStep (0, 1, 2, 3) và gán các phép toán stream cho bốn bước CStep (0, 1, 2, 3) như sau:

- CStep 0: Một phép toán $split$ và năm phép toán $merge$ gán cho bước CStep 0.
- CStep 1: Một phép toán $drop$, một phép toán zip, một phép toán rev và một phép toán $take$ gán cho bước CStep 1.
- CStep 2: Ba phép toán $merge$ gán cho bước CStep 2 và
- CStep 3: Ba phép toán $merge$ gán cho bước CStep 3

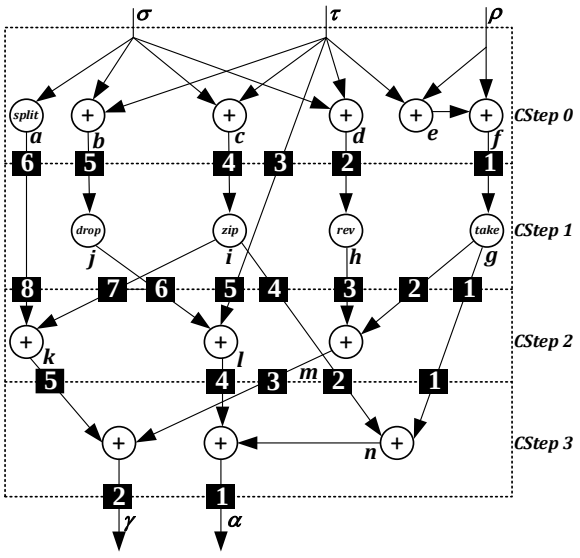
Trong thực tế, việc lập lịch cho tính toán BDL dựa trên phép toán stream ở trên không chỉ có đó mà sau đó có việc lập lịch BDL khác sắp xếp một cách rõ ràng. Do đó, hai yếu tố chính sau đây phải được xem xét và ước lượng một cách phù hợp.

- Cần bao nhiêu bước CStep là thích hợp?
- Các yêu cầu về phần cứng cần bao nhiêu để triển khai các phép toán stream?

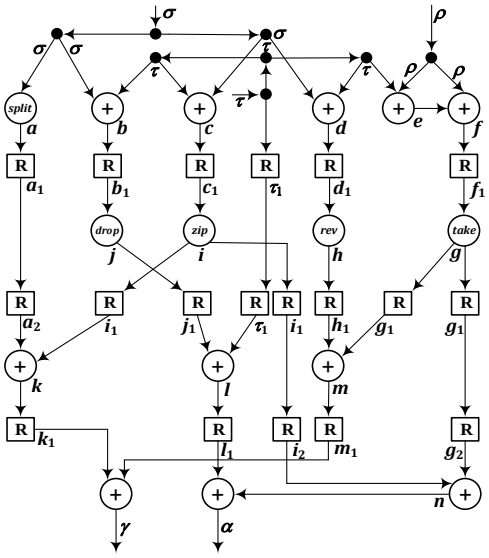
Hai yếu tố trên là rất quan trọng vì số lượng CStep xác định tốc độ triển khai, trong khi các yêu cầu tài nguyên phần cứng xác định kích thước của việc triển khai. Hơn nữa, một tiến trình lập lịch BDL cần xem xét với ba thách thức sau đây:

- Các ràng buộc yêu cầu tài nguyên phần cứng cho các phép toán stream.
- Các ràng buộc về thời gian cho hiệu năng.
- Để giải quyết thách thức lập lịch, yêu cầu các thuật toán lập lịch BDL phải phù hợp.

3.2.2.3. Cấp phát và liên kết thanh ghi



Hình 3.5. Sơ đồ cấp phát và liên kết thanh ghi



Hình 3.6. Sơ đồ tính toán stream dựa trên phép toán stream biểu diễn cho RTL

Trong cấp phát và liên kết thanh ghi bao gồm bước tổng hợp lập trình stream

thực thi hai nhiệm vụ như sau:

- Cấp phát là xác định số lượng thanh ghi được yêu cầu để lưu trữ các kết quả trung gian giữa hai CStep.
- Liên kết là ánh xạ các kết quả trung gian đến các thanh ghi cho mỗi CStep.

Trong cấp phát thanh ghi yêu cầu rằng số lượng thanh ghi phải bằng với số lượng lớn nhất của kết quả trung gian giữa hai CStep. Trong hình 3.5, ta cần sáu thanh ghi được đánh số là 1, 2, 3, 4, 5, 6 để cấp phát và liên kết giữa CStep 0 và CStep 1, các thanh ghi 1, 2, 3, 4, 5 và 6 được sử dụng lại giữa CStep 1 và CStep 2, giữa CStep 2 và CStep 3, các thanh ghi 1, 2, 3, 4 và 5 lại được sử dụng.

Quá trình cấp phát và liên kết thanh ghi có ảnh hưởng đến việc sử dụng các yêu cầu tài nguyên phần cứng. Nói một cách khác, các thanh ghi có thể được cấp phát và liên kết thì càng tốt, yêu cầu tài nguyên phần cứng bổ sung càng được loại bỏ. Đối với $\sigma, \tau, \rho, \alpha, \gamma \in \mathbb{A}^{\omega}$, các stream trong hình 3.5 và áp dụng định lý 2.1 (Chương 2) để thu được các phương trình vi phân như sau:

- $a = \sigma, b = \sigma + \tau, c = \sigma + \tau, d = \sigma + \tau, e = \tau + \rho$
- $f = e + \rho = (\tau + \rho) + \rho = \tau + \rho + \rho = \tau + 2\rho$ (ở đó 2ρ là merge tách rời)
- $g = X \times f = X \times (\tau + 2\rho)$
- $h = X \times d = X \times (\sigma + \tau)$
- $i = X \times c = X \times (\sigma + \tau)$
- $j = X \times b = X \times (\sigma + \tau)$
- $k = X \times X \times a + X \times i = X^2 \times \sigma + X \times \big(X \times (\sigma + \tau)\big) = X^2 \times \sigma + \big(X^2 \times (\sigma + \tau)\big) = X^2 \times \big(\sigma + (\sigma + \tau)\big) = X^2 \times (2\sigma + \tau)$
- $l = X \times j + X \times X \times \tau = X \times j + X^2 \times \tau = X \times \big(X \times (\sigma + \tau)\big) + X^2 \times \tau = X^2 \times (\sigma + \tau) + X^2 \times \tau = X^2 \times \big((\sigma + \tau) + \tau\big) = X^2 \times (\sigma + 2\tau)$
- $m = X \times h + X \times g = X \times (h + g) = X \times \Big(\big(X \times (\sigma + \tau)\big) + \big(X \times (\tau + 2\rho)\big)\Big) = X^2 \times \big((\sigma + \tau) + (\tau + 2\rho)\big) = X^2 \times (\sigma + 2\tau + 2\rho)$
- $n = X \times X \times i + X \times X \times g = X^2 \times (i + g) = X^2 \times \Big(\big(X \times (\sigma + \tau)\big) + \big(X \times (\tau + 2\rho)\big)\Big) = X^3 \times (\sigma + 2\tau + 2\rho)$

Do đó, ta có kết quả stream α đầu ra dạng BDL như sau: $\alpha = X \times l + n = X \times l + X^2 \times (i + g) = X \times \big(X^2 \times (\sigma + 2\tau)\big) + X^2 \times \Big(\big(X \times (\sigma + \tau)\big) + \big(X \times (\tau + 2\rho)\big)\Big) = \big(X^3 \times (\sigma + 2\tau)\big) + \big(X^3 \times (\sigma + 2\tau + 2\rho)\big) = X^3 \times (2\sigma + 4\tau + 2\rho)$ và kết quả stream γ đầu ra dạng BDL như sau: $\gamma = X \times k + X \times m = X \times \big(X^2 \times (2\sigma + \tau)\big) + X \times \big(X^2 \times (\sigma + 2\tau + 2\rho)\big) = X^3 \times (3\sigma + 3\tau + 2\rho)$.

Dựa vào tính toán các phương trình trên, đặt hai stream μ và δ như sau $\mu = (2\sigma + 4\tau + 2\rho)$ và $\delta = (3\sigma + 3\tau + 2\rho)$, do đó tạo ra các stream đầu ra được ký hiệu như sau $\alpha = X^3 \times \mu$ và $\gamma = X^3 \times \delta$. Qua quá trình tính toán ta có được stream μ và δ từ các stream σ, τ và ρ, và sau đó sử dụng chúng để tạo ra các stream đầu ra α và

γ thông qua các phương trình vi phân hành vi. Như vậy, phương trình vi phân hành vi ở bảng 3.1 biểu diễn cho hai stream đầu ra α và γ.

Bảng 3.1. Phương trình vi phân hành vi biểu diễn stream α và γ

Phương trình vi phân stream	Giá trị ban đầu
$\alpha^{(3)} = \mu$ $\gamma^{(3)} = \delta$	$\alpha(0) = \alpha(1) = \alpha(2)$ $\gamma(0) = \gamma(1) = \gamma(2)$

Sơ đồ stream biểu diễn trong hình 3.6 cũng có thể được xem như một sơ đồ stream của RTL trong hình 3.5. Tổng quát hơn, nếu tính toán được lập lịch trong n CStep, $n \geq 0$ thì stream đầu ra α và γ dạng BDL được xác định bởi phương trình sau: $\alpha = X^n \times \mu$ và $\gamma = X^n \times \delta$. Như vậy, phương trình vi phân hành vi được mô tả ở bảng 3.2 biểu diễn cho hai stream đầu ra α và γ.

Bảng 3.2. Phương trình vi phân hành vi biểu diễn cho stream α và γ đầu ra

Phương trình vi phân stream	Giá trị ban đầu
$\alpha^{(n)} = \mu$ $\gamma^{(n)} = \delta$	$\alpha^{(i)} = \mu(0), 0 \leq i \leq n-1$ $\gamma^{(i)} = \delta(0), 0 \leq i \leq n-1$

Một quan hệ mô phỏng hai chiều B trên $\mathbb{A}^\omega$ là một quan hệ $B \subseteq \mathbb{A}^\omega \times \mathbb{A}^\omega$ sao cho $\forall \varphi, \beta \in \mathbb{A}^\omega$, nếu $\varphi \; B \; \beta$ thì $\varphi(0) = \beta(0)$ và $\varphi' \; B \; \beta'$ *(Tiểu mục 1.6.5, Định nghĩa 1.4, Chương 1)*.

Mệnh đề 3.1 (Mô phỏng hai chiều): $\forall n \geq 0$, tất cả các stream đầu ra dạng BDL $\alpha = X^n \times \mu$ (hoặc $\gamma = X^n \times \delta$) đều là mô phỏng hai chiều theo cặp với stream α (hoặc với stream γ).

Chứng minh: Xét một stream $\alpha = X^n \times \mu$. Ta định nghĩa một quan hệ mô phỏng hai chiều $B \subseteq \mathbb{A}^\omega \times \mathbb{A}^\omega$ gồm các cặp stream con của α có dạng: $B = \{\langle X^{i+k} \times \mu \, , X^{j+k} \times \mu \rangle \mid 0 \leq i, j \leq n, k \geq 0\}$. Mỗi cặp trong B biểu diễn một mô phỏng hai chiều vì phần đầu $X^{i+0} \times \mu \; = X^{j+0} \times \mu$ và phần đuôi (đạo hàm) của hai stream con $X^{i+k} \times \mu \; B \; X^{j+k} \times \mu$ sau mỗi bước dịch chuyển k. Như vậy, mọi stream con dạng $X^n \times \mu$ đều mô phỏng lẫn nhau hai chiều. □

Hệ quả 3.1 (Đồng quy nạp): Với mọi stream đầu ra $X^n \times \mu$ trong $\mathbb{A}^\omega$, nếu $X^i \times \mu \sim X^j \times \mu$ $(0 \leq i, j \leq n)$, thì $X^i \times \mu = X^j \times \mu$. Nói cách khác, nếu hai stream đầu ra $X^i \times \mu$ và $X^j \times \mu$ là tương đương $\sim$, thì $X^i \times \mu$ và $X^j \times \mu$ cũng bằng nhau.

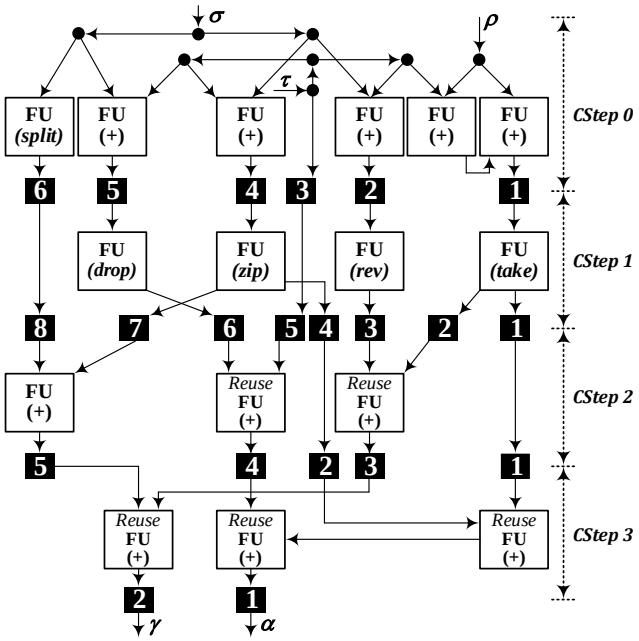
Chứng minh: Xét hai stream đầu ra $\alpha_1 = X^i \times \mu$ và $\alpha_2 = X^j \times \mu$. Theo mệnh đề 3.1, tồn tại quan hệ mô phỏng hai chiều $B \subseteq \mathbb{A}^\omega \times \mathbb{A}^\omega$ sao cho $\langle \alpha_1, \alpha_2 \rangle \in B$, và dựa vào đồng quy nạp trên k $(k \geq 0)$ *(Tiểu mục 1.6.5, Định lý 1.2, Chương 1)*, ta có $\langle \alpha_1^{i+k}, \alpha_2^{j+k} \rangle \in B$, vì B là một mô phỏng hai chiều nên do đó $\alpha_1^{i+k}(0) = \alpha_2^{j+k}(0)$. Tức là mọi thành phần trong hai stream đều bằng nhau. Do đó, $\alpha_1 = \alpha_2$. □

3.2.2.4. Cấp phát và liên kết đơn vị chức năng

Trong việc cấp phát và liên kết các đơn vị chức năng, ở đó một đơn vị chức

năng (FU) được thực hiện hai nhiệm vụ như sau:

- *Cấp phát là xác định các phép toán stream để triển khai cho mỗi CStep.*
- *Liên kết là triển khai các phép toán stream trên các sơ đồ stream.*



Hình 3.7. Sơ đồ cấp phát và liên kết các FUs

Các đơn vị chức năng (FUs) thường được cấu trúc trong một thư viện biểu diễn cho sự ánh xạ giữa các chức năng của nó và các phép toán stream liên quan. FUs có thể triển khai các phép toán stream đơn dụng hoặc các phép toán đa dụng chức năng với các tín hiệu đầu vào điều khiển để lựa chọn được phép toán stream mong muốn. Trong *hình 3.7*, tám FUs đơn dụng chức năng được yêu cầu để triển khai các phép toán stream: $drop, take, zip, split, rev, +, \bullet$, và ■ theo yêu cầu. Đơn dụng, mỗi FU đơn dụng được thiết kế cho một phép toán duy nhất (các phép toán xử lý số học: cộng, nhân, chia, so sánh, các phép toán xử lý stream: $drop$, $take$, zip, $split$, rev, $merge$ $+$, $\bullet$, và ■, v.v.). Không tái cấu hình, khác với FU đa dụng, các FU đơn dụng không thể thay đổi để thực hiện phép toán khác ngoài phép toán đã được thiết kế. Tối ưu về tốc độ, vì được chuyên biệt hóa, FU đơn dụng thường có độ trễ thấp, hoạt động nhanh và chính xác cho phép toán đó. Ứng dụng trong lập lịch pipeline, khi xây dựng lịch pipeline, ta có thể gán từng FU đơn dụng cho một phép toán stream cụ thể ở mỗi CStep, đảm bảo tính song song và giảm độ phức tạp của việc điều khiển. Bảng 3.3 giúp làm rõ ưu và nhược điểm trong bối cảnh tối ưu hóa lập lịch theo pipeline, nếu cần hiệu năng tối đa thì chọn FU đơn dụng, còn nếu cần linh hoạt và tiết kiệm tài nguyên thì chọn FU đa dụng.

Bảng 3.3. So sánh FU đơn dụng và FU đa dụng

Tiêu chí	FU đơn dụng	FU đa dụng
Chức năng	Chỉ thực hiện một phép toán duy nhất (Ví dụ: $drop$, $take$, zip, $split$, rev,	Có thể thực hiện nhiều phép toán khác nhau nhờ mạch điều

Tiêu chí	FU đơn dụng	FU đa dụng
	$merge$ +, •, ■, v.v).	khiển.
Thiết kế	Đơn giản, tối ưu cho một nhiệm vụ cụ thể.	Phức tạp hơn do cần bộ điều khiển để lựa chọn chức năng.
Tốc độ và độ trễ	Nhanh, độ trễ thấp vì chuyên biệt hóa.	Có thể chậm hơn do phải chuyển đổi chế độ hoạt động.
Tính linh hoạt	Hạn chế, không thể tái sử dụng cho phép toán khác.	Dễ thích ứng cho nhiều loại tác vụ khác nhau.
Tài nguyên phần cứng	Tốn nhiều phần cứng nếu số lượng phép toán cần thực hiện đa dạng.	Tiết kiệm phần cứng nhờ dùng chung cho nhiều phép toán.
Ứng dụng điển hình	Xử lý tín hiệu, hệ thống chuyên dụng, cơ chế đường ống (pipeline) cố định.	Bộ xử lý tổng quát, kiến trúc tái cấu hình, môi trường linh hoạt.

3.2.2.5. Bài toán tối ưu hóa lập lịch theo cơ chế đường ống tính toán stream

Đặt bài toán: Cho biết 3 stream $\sigma, \tau, \rho \in \mathbb{A}^{\omega}$ đầu vào hai biểu thức đại số stream của thuật toán 3.1: $\alpha := \Big(zip(\sigma + \tau) + take\big(\rho + (\tau + \rho)\big) + (drop(\sigma + \tau) + \tau)\Big)$ và $\gamma := \Big(split(\sigma) + zip(\sigma + \tau) + \Big(rev(\sigma + \tau) + take\big(\rho + (\tau + \rho)\big)\Big)\Big)$. Luận án đề xuất kịch bản sau đây:

- **Kịch bản:** Tối ưu hóa lập lịch giảm độ trễ (qua việc chia CStep), tăng thông lượng (nhờ cơ chế đường ống - pipeline), cân bằng tải (nhờ phân hoạch đều các phép toán stream trên từng CStep), và tối ưu tài nguyên (nhờ tái sử dụng thanh ghi và giá trị trung gian) theo cơ chế đường ống dựa trên quy hoạch tuyến tính khi biết hai biểu thức đại số stream. Tức là, cho biết hai biểu thức đại số stream, luận án tối ưu hóa lập lịch, với 3 stream σ, τ, ρ đầu vào.
- **Trình tự sự kiện:** Thực hiện thuật toán lập lịch 3.1 với 13 bước điều khiển (CStep) theo ràng buộc phụ thuộc dữ liệu của mô hình LP-Based PSOM [CT.9].
- **Chỉ số quan sát được:** Trích xuất giá trị tối ưu $[S_1 - S_{13}]$ = 5 CStep, tổng độ trễ (C_{max}) = 5 CStep, sơ đồ tối ưu hóa lập lịch theo cơ chế đường ống (Hình 3.10).
- **Kết quả kỳ vọng:** Giảm số bước điều khiển (CStep) từ 13 xuống còn 5 CStep, thông lượng tăng lên khoảng 2.5 lần, tài nguyên được tái sử dụng hiệu quả hơn.
- **Môi trường giả lập:** colab.research.google.com và ngôn ngữ lập trình Python.

a) Mô tả các phép toán xử lý stream dạng BDL của thuật toán 3.1

Bảng 3.4. Mô tả các phép toán xử lý stream dùng trong *thuật toán 3.1*

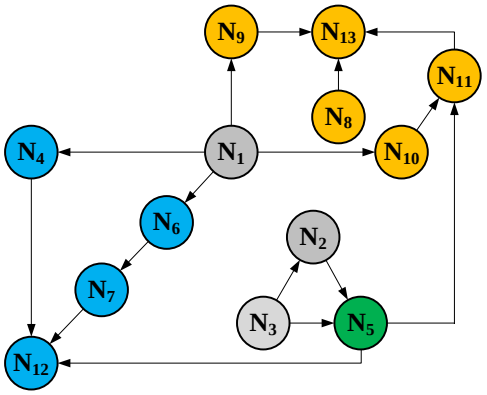
Phép toán	Mô tả chi tiết các phép toán xử lý stream
$zip(\sigma, \tau)$	Nén từng cặp frame từ 2 stream σ và τ
$take(\sigma)$	Lấy frame đầu tiên của stream σ
$drop(\sigma)$	Bỏ frame đầu tiên của stream σ
$split(\sigma)$	Tách stream σ thành 2 phần
$rev(\sigma)$	Đảo ngược thứ tự các frame trong stream σ
$\sigma + \tau$	Trộn (+) hai stream σ và τ theo thứ tự
$:=$	Gán giá trị stream với stream

b) Mô hình hóa từng phép toán xử lý stream thành nút trên sơ đồ

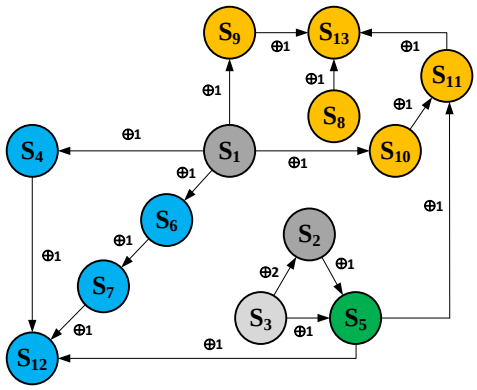
Bảng 3.5. Mô hình hóa các phép toán xử lý stream cho *thuật toán 3.1*

Nút N_i	Biến S_i	Phép toán xử lý stream	Thuộc nhánh α và γ
N_1	S_1	$merge_1 := \sigma + \tau$	Dùng chung
N_2	S_2	$merge_2 := \rho + (\tau + \rho)$	
N_3	S_3	$merge_3 := \tau + \rho$	Trung gian
N_4	S_4	$zip_1 := zip(merge_1)$	α
N_5	S_5	$take_1 := take(merge_2)$	α và γ
N_6	S_6	$drop_1 := drop(merge_1)$	α
N_7	S_7	$merge_4 := drop_1 + \tau$	
N_8	S_8	$split_1 := split(\sigma)$	γ
N_9	S_9	$zip_2 := zip(merge_1)$	
N_{10}	S_{10}	$rev_1 := rev(merge_1)$	
N_{11}	S_{11}	$merge_5 := rev_1 + take_1$	
N_{12}	S_{12}	$merge_6 := zip_1 + take_1 + merge_4$	α
N_{13}	S_{13}	$merge_7 := split_1 + zip_2 + merge_5$	γ

Dựa vào bảng 3.4 và 3.5, luận án mô hình hóa sự phụ thuộc của các phép toán xử lý stream trong *thuật toán 3.1* thành sơ đồ các nút từ N_1 đến N_{13} (Hình 3.8) và sau đó được sơ đồ hóa thành các biến từ S_1 đến S_{13} tương ứng (Hình 3.9).



Hình 3.8. Mô hình hóa từng phép toán xử lý stream thành một nút trên sơ đồ



Hình 3.9. Mô hình hóa từng phép toán xử lý stream thành một biến số trên sơ đồ

c) Mô hình hóa bài toán quy hoạch tuyến tính

Xác định ràng buộc phụ thuộc và ràng buộc không chồng lắp dữ liệu giữa các phép toán stream trong pipeline. Mỗi nút xử lý N_i (Hình 3.8) được gán cho biến $S_i \in \mathbb{Z}^+$ (Bảng 3.5, Hình 3.9) đại diện cho CStep mà phép toán N_i thực thi. Gọi C là bước điều khiển tối đa (CStep) và là độ trễ tối đa của cơ chế đường ống cần tối thiểu hóa. Vậy ta có hàm mục tiêu để tối thiểu hóa độ trễ cuối (số bước tối đa trong pipeline): Tối thiểu hóa C_{max}, với ràng buộc $C_{max} \geq S_i, \forall i \in \mathbb{Z}^+$.

- **Phân tích tổng quát sự phụ thuộc giữa các phép toán stream:** Giả sử S_i và S_j là thời gian thực hiện phép toán thứ i và j (CStep). Ràng buộc dữ liệu giữa nút i và j phải đảm bảo rằng $S_i \geq S_j \oplus \text{Delay}$ và $|S_i - S_j| \geq 1$, với $1 \leq i, j \leq n$ (Hình 3.9), trong đó:
  - $\oplus$ là phép toán cộng số nguyên dương $\mathbb{Z}^+$ và cũng được sử dụng để phân biệt nó với phép toán merge (+). Số n là tổng số phép toán trong thuật toán

- Nút phụ thuộc là nút i. Nút chia sẻ (nút trung gian) là nút j
- Delay là đơn vị độ trễ và là tham số tùy chỉnh tùy ý. Các nhãn $\oplus 1$, $\oplus 2$, v.v. trên mỗi cạnh chỉ số lượng CSteps phải chờ (Hình 3.9).
- Thể hiện không vi phạm ràng buộc sử dụng dữ liệu trước khi có dữ liệu
- Cho phép lập lịch theo cơ chế đường ống hợp lệ và hiệu quả
- Luận án xác định ràng buộc phụ thuộc và ràng buộc không chồng lắp giữa các nút theo phương pháp quy hoạch tuyến tính như sau (Bảng 3.6).

Bảng 3.6. Xác định ràng buộc phụ thuộc và ràng buộc không chồng lắp dữ liệu các nút

Ràng buộc phụ thuộc và ràng buộc không chồng lắp giữa nút N_i và N_j	**Biểu thức quy hoạch tuyến tính ($S_i \geq S_j \oplus$ Delay và $\lvert S_{i,j} - S_{j,i} \rvert \geq 1$)**
N_4 phụ thuộc N_1	$S_4 \geq S_1 \oplus 1$
N_5 phụ thuộc N_2 và N_3	$S_5 \geq \max(S_2 \oplus 1, S_3 \oplus 2)$
N_2 phụ thuộc N_3	$S_2 \geq S_3 \oplus 1$
N_6 phụ thuộc N_1	$S_6 \geq S_1 \oplus 1$
N_7 phụ thuộc N_6	$S_7 \geq S_6 \oplus 1$
N_9 phụ thuộc N_1	$S_9 \geq S_1 \oplus 1$
N_{10} phụ thuộc N_1	$S_{10} \geq S_1 \oplus 1$
N_{11} phụ thuộc N_{10} và N_5	$S_{11} \geq \max(S_{10} \oplus 1, S_5 \oplus 1)$
N_{12} phụ thuộc N_4, N_5 và N_7	$S_{12} \geq \max(S_4 \oplus 1, S_5 \oplus 1, S_7 \oplus 1)$
N_{13} phụ thuộc N_8, N_9 và N_{11}	$S_{13} \geq \max(S_8 \oplus 1, S_9 \oplus 1, S_{11} \oplus 1)$
N_4 không chồng lắp N_9	$\lvert S_4 - S_9 \rvert \geq 1$
N_9 không chồng lắp N_{10}	$\lvert S_9 - S_{10} \rvert \geq 1$
N_6 không chồng lắp N_{10}	$\lvert S_6 - S_{10} \rvert \geq 1$
N_6 không chồng lắp N_9	$\lvert S_6 - S_9 \rvert \geq 1$
N_4 không chồng lắp N_{10}	$\lvert S_4 - S_{10} \rvert \geq 1$

- **Phân tích cụ thể sự phụ thuộc giữa các phép toán stream:** Trong ngữ cảnh tối ưu hóa lập lịch theo cơ chế đường ống bằng phương pháp quy hoạch tuyến tính, biểu thức quy hoạch tuyến tính trong bảng 3.6 sẽ là:
  - S_1 (nghĩa là S_j) là thời điểm mà nút xử lý N_1 là $merge_1 := \sigma + \tau$ được thực hiện. Trong khi, $S_1 \oplus 1$ là bước xử lý ngay sau khi nút N_1 hoàn thành, tức là S_4 (nghĩa là S_i) của nút N_4 chỉ được thực hiện ở bước tiếp theo trở đi vì phụ thuộc đầu ra của S_1.
  - S_3 (nghĩa là S_j) là thời điểm mà nút xử lý N_3 là $merge_3 := \tau + \rho$ được thực hiện. Vì đơn vị độ trễ giữa S_3 so với S_5 là 1, còn đơn vị độ trễ giữa S_3 so với S_2 là 2 nên chọn max là 2, nên $S_3 \oplus 2$ là bước xử lý ngay sau khi nút N_3 hoàn thành, tức là S_2 (nghĩa là S_i) của nút N_2 chỉ được thực hiện ở bước tiếp theo trở đi vì phụ thuộc đầu ra của S_3. Tiếp theo, S_2 (nghĩa là S_j) là thời điểm mà nút xử lý N_2 là $merge_2 := \rho + (\tau + \rho)$ được thực hiện. Trong khi, $S_2 \oplus 1$ là bước xử lý ngay sau khi nút N_2 hoàn thành, tức là S_5 của nút N_5 chỉ được thực hiện ở bước tiếp theo trở đi vì phụ thuộc đầu ra của S_2.
  - S_1 (nghĩa là S_j) là thời điểm mà nút xử lý N_1 là $merge_1 := \sigma + \tau$ được thực hiện. Trong khi, $S_1 \oplus 1$ là bước xử lý ngay sau khi nút N_1 hoàn thành, tức là S_6 (nghĩa là S_i) của nút N_6 chỉ được thực hiện ở bước tiếp theo trở đi vì phụ thuộc đầu ra của S_1.

- S_6 (nghĩa là S_j) là thời điểm mà nút xử lý N_6 là $drop_1 := drop(merge_1)$ được thực hiện. Trong khi, $S_6 \oplus 1$ là bước xử lý ngay sau khi N_6 hoàn thành, tức là S_7 (nghĩa là S_i) của nút N_7 chỉ được thực hiện ở bước tiếp theo trở đi vì phụ thuộc đầu ra của S_6.
- S_1 (nghĩa là S_j) là thời điểm mà nút xử lý N_1 là $merge_1 := \sigma + \tau$ được thực hiện. Trong khi, $S_1 \oplus 1$ là bước xử lý ngay sau khi nút N_1 hoàn thành, tức là S_9 (nghĩa là S_i) của nút N_9 chỉ được thực hiện ở bước tiếp theo trở đi vì phụ thuộc đầu ra của S_1.
- S_1 (nghĩa là S_j) là thời điểm mà nút xử lý N_1 là $merge_1 := \sigma + \tau$ được thực hiện. Trong khi, $S_1 \oplus 1$ là bước xử lý ngay sau khi nút N_1 hoàn thành, tức là S_{10} (nghĩa là S_i) của nút N_{10} chỉ được thực hiện ở bước tiếp theo trở đi vì phụ thuộc đầu ra của S_1.
- S_{10} (nghĩa là S_j) là thời điểm mà nút xử lý N_{10} là $rev_1 := rev(merge_1)$ được thực hiện song song với nút xử lý N_5 là $take_1 := take(merge_2)$. Trong khi, $S_{10} \oplus 1$ và $S_5 \oplus 1$ là bước xử lý ngay sau khi nút N_{10} và N_5 hoàn thành, tức là S_{11} (nghĩa là S_i) của nút N_{11} chỉ được thực hiện bước tiếp theo trở đi vì phụ thuộc đầu ra của S_{10} và S_5.

d) Xây dựng mô hình tối ưu hóa lập lịch theo cơ chế đường ống

Mô hình tối ưu hóa lập lịch theo cơ chế đường ống (LP-Based PSOM) được đặc tả bằng các bước điều khiển (CStep) mà các biến S_k tại nút thứ k được thực thi, $\forall k \in \mathbb{Z}^+$. Mục đích là tối thiểu hóa số bước điều khiển cuối $\max(\{S_1, S_2, \dots, S_{13}\})$, hàm mục tiêu là tối thiểu hóa C (Độ trễ tối đa C_{max}), với $C_{max} \geq S_k$, $\forall k \in \mathbb{Z}^+$, và ràng buộc không chồng lắp $|S_i - S_j| \geq 1$ hoặc $|S_j - S_i| \geq 1$.

Các ràng buộc không chồng lắp gồm: $|S_4 - S_9| \geq 1$, $|S_9 - S_{10}| \geq 1$, $|S_6 - S_{10}| \geq 1$, $|S_6 - S_9| \geq 1$, và $|S_4 - S_{10}| \geq 1$ đảm bảo có thể truy xuất đồng thời nhưng không xung đột từ $merge_1$. Về hiệu quả, tránh thắt cổ chai trong pipeline và giúp hệ thống hoạt động tốt. Từ đây, luận án phân tích có 29 ràng buộc phụ thuộc và 5 ràng buộc không chồng lấp để xây dựng mô hình LP-based PSOM như sau:

$$\begin{cases} C_{max} \geq S_k, \forall k = 1, 2, 3, \dots, 13 \\ S_4 \geq S_1 \oplus 1 \\ S_5 \geq S_2 \oplus 1, S_5 \geq S_3 \oplus 2 \\ S_2 \geq S_3 \oplus 1 \\ S_6 \geq S_1 \oplus 1 \\ S_7 \geq S_6 \oplus 1 \\ S_9 \geq S_1 \oplus 1 \\ S_{10} \geq S_1 \oplus 1 \\ S_{11} \geq S_{10} \oplus 1, S_{11} \geq S_5 \oplus 1 \\ S_{12} \geq S_4 \oplus 1, S_{12} \geq S_5 \oplus 1, S_{12} \geq S_7 \oplus 1 \\ S_{13} \geq S_8 \oplus 1, S_{13} \geq S_9 \oplus 1, S_{13} \geq S_{11} \oplus 1 \\ |S_4 - S_9| \geq 1 \\ |S_9 - S_{10}| \geq 1 \\ |S_6 - S_{10}| \geq 1 \\ |S_6 - S_9| \geq 1 \\ |S_4 - S_{10}| \geq 1 \end{cases}$$

e) Tối ưu hóa lập lịch theo cơ chế đường ống

Các ràng buộc *mục d*, luận án xây dựng bảng 3.7 tối ưu hóa lập lịch theo cơ chế đường ống cho stream dạng BDL cho thuật toán 3.1 với 5 CStep (C) như sau:

Bảng 3.7. Phân chia các CStep cho các phép toán xử lý stream được thực hiện

C	Phép toán được thực hiện	Số phép toán	Mô tả phép toán	Nhánh	Ghi chú tài nguyên
C_1	$(S_1)merge_1 := \sigma + \tau$, $(S_3)merge_3 := \tau + \rho$, $(S_8)split_1 := split(\sigma)$	3	Tạo các giá trị trung gian dùng chung.	Dùng chung, α, γ	Chuẩn bị dữ liệu trung gian
C_2	$(S_2)merge_2 := \rho + merge_3$, $(S_{10})rev_1 := rev(merge_1)$	2	Các phép toán phụ thuộc $merge_1$ và $merge_3$	α, γ	Khai thác phụ thuộc, dùng rev
C_3	$(S_4)zip_1 := zip(merge_1)$, $(S_5)take_1 := take(merge_2)$, $(S_6)drop_1 := drop(merge_1)$	3	Dùng lại $merge_1$ song song tại 2 nhánh α và γ	Dùng chung α, γ	Tái sử dụng $merge_1$ và dùng chung $merge_2$
C_4	$(S_7)merge_4 := drop_1 + \tau$, $(S_9)zip_2 := zip(merge_1)$ $(S_{11})merge_5 := rev_1 + take_1$	3	Phép toán phụ thuộc $merge_1$, kết hợp dữ liệu tại nhánh α, γ	α, γ	Kết hợp dữ liệu từ C_2, C_3
C_5	$(S_{12})merge_6 := zip_1 + take_1 + merge_4$, $(S_{13})merge_7 := split_1 + zip_2 + merge_5$	2	Trộn các nhánh stream	α, γ	Phép toán trộn nhánh α và γ

Mô hình tối ưu hóa lập lịch theo cơ chế đường ống dựa trên quy hoạch tuyến tính đã đạt được bốn mục tiêu cốt lõi như sau:

- ***Giảm độ trễ tổng thể:*** Trên thực tế nếu xử lý lập lịch tuần tự 13 phép toán thì cần 13 CStep. Với cơ chế đường ống, nhờ chia thành 5 CStep nên độ trễ từ đầu đến cuối giảm chỉ còn dưới một nửa. Về hiệu quả, hệ thống phản hồi nhanh hơn trong bối cảnh livestream thời gian thực.
- ***Tăng thông lượng nhờ song song hóa:*** Các phép toán như $zip_1 := zip(merge_1)$, $drop_1 := drop(merge_1)$, $take_1 := take(merge_2)$ (C_3) không phụ thuộc nhau nên được thực hiện xử lý song song và dùng chung. Các phép toán $merge_4 := drop_1 + \tau$ và $merge_5 := rev_1 + take_1$ có thể diễn ra đồng thời tại C_4. Về hiệu quả, số lượng phép toán thực thi đồng thời tăng, giảm thời gian chờ.
- ***Tối ưu tài nguyên hệ thống (tái sử dụng thanh ghi):*** Phép toán $merge_1 := \sigma + \tau$ được dùng lại cho phép toán $zip_1 := zip(merge_1)$, $drop_1 := drop(merge_1)$, $zip_2 := zip(merge_1)$, và $rev_1 := rev(merge_1)$ trong nhánh α và γ. Phép toán $merge_3 := \tau + \rho$ dùng lại trong $merge_2$, từ đó được sử dụng trong $take_1 := take(merge_2)$. Về hiệu quả, giảm yêu cầu bộ nhớ hay tái tạo dữ liệu nên tiết kiệm thanh ghi.
- ***Cân bằng tải và tránh xung đột:*** Do phân bố đều từ 2 đến 3 phép toán stream trên từng CStep nên cân bằng xử lý. Các ràng buộc không chồng lắp gồm $|S_4 - S_9| \geq 1$, $|S_9 - S_{10}| \geq 1$, $|S_6 - S_{10}| \geq 1$, $|S_6 - S_9| \geq 1$, và $|S_4 - S_{10}| \geq 1$ đảm bảo có thể truy

xuất đồng thời nhưng không xung đột từ $merge_1$. Về hiệu quả, tránh thắt cổ chai trong pipeline, giúp hệ thống hoạt động tốt.

f) Hiện thực mô hình tối ưu hóa lập lịch theo cơ chế đường ống

Thuật toán 3.2: Tối ưu hóa lập lịch pipeline dựa trên quy hoạch tuyến tính [CT.9]

Đầu vào: Ba stream $\sigma, \tau, \rho \in \mathbb{A}^{\omega}$; Các biến lập lịch $S = \{S_1, S_2, \ldots, S_{13}\}$

Đầu ra : Trạng thái bài toàn (Tối ưu); Tổng độ trễ ($C_{\max}$); Sơ đồ tối ưu hóa lập lịch pipeline

1. Xác định bài toán LP dưới dạng bài toán tối thiểu hóa: Giảm thiểu C (CStep lớn nhất)
2. For $i = 1$ to 13
3. Xác định biến nguyên $S_i \geq 1$
4. Xác định biến nguyên $C \geq 1$
5. Đặt hàm mục tiêu: Giảm thiểu C
6. For $i = 1$ to 13
7. Thêm ràng buộc độ trễ: $C \geq S_i$
8. *Thêm các ràng buộc thứ tự:* 8.1. $S_4 \geq S_1 \oplus 1$, 8.2. $S_5 \geq S_2 \oplus 1$, 8.3. $S_5 \geq S_3 \oplus 2$, 8.4. $S_2 \geq S_3 \oplus 1$, 8.5. $S_6 \geq S_1 \oplus 1$, 8.6. $S_7 \geq S_6 \oplus 1$, 8.7. $S_9 \geq S_1 \oplus 1$, 8.8. $S_{10} \geq S_1 \oplus 1$, 8.9. $S_{11} \geq S_{10} \oplus 1$, 8.10. $S_{11} \geq S_5 \oplus 1$, 8.11. $S_{12} \geq S_4 \oplus 1$, 8.12. $S_{12} \geq S_5 \oplus 1$, 8.13. $S_{12} \geq S_7 \oplus 1$, 8.14. $S_{13} \geq S_8 \oplus 1$, 8.15. $S_{13} \geq S_9 \oplus 1$, 8.16. $S_{13} \geq S_{11} \oplus 1$
9. Với mỗi cặp tác vụ $(i, j) \in \{(4,9), (6,10), (9,10), (6,9), (4,10)\}$:
   - 9.1. Đưa vào biến nhị phân $b_{ij} \in \{0,1\}$
   - 9.2. Áp dụng ràng buộc big-M: $S_i - S_j \geq 1 - M * b_{ij}$
   - 9.3. Áp dụng ràng buộc big-M: $S_j - S_i \geq 1 - M * (1 - b_{ij})$
10. Giải bài toán LP
11. *Xuất trạng thái bài toán (Tối ưu hóa):* Trích xuất giá trị S_1 đến S_{13}
12. Tính tổng độ trễ $C_{\max}$
13. Ánh xạ từng tác vụ vào nhánh tương ứng (α, γ, trung gian, chia sẻ) (Hình 3.10)
14. Trực quan hóa lịch dưới dạng sơ đồ tối ưu hóa lập lịch pipeline (Hình 3.10)

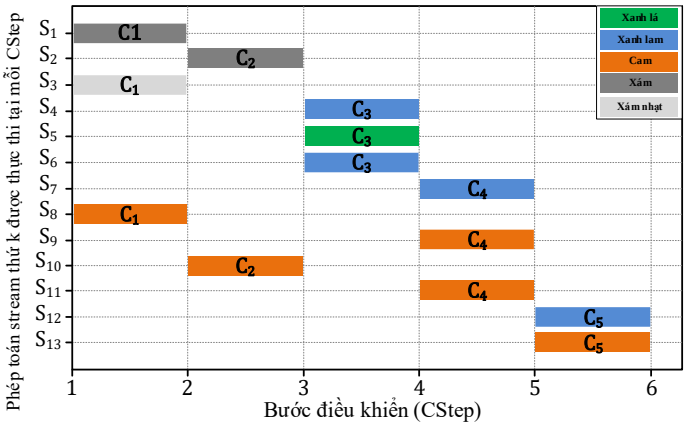
Lập lịch theo cơ chế đường ống là một tiếp cận lý tưởng cho xử lý stream dạng BDL trong livestream thời gian thực, nơi tính trễ thấp và hiệu quả về tài nguyên là yêu cầu bắt buộc. Lập lịch theo pipeline đã rút ngắn độ trễ từ đầu đến cuối: 5 CStep thay vì 13 CStep, tăng thông lượng nhờ song song hóa, giảm tài nguyên nhờ tái sử dụng thanh ghi, và cân bằng tải giữa các bước và tránh xung đột truy cập.

Luận án hiện thực LP-Based PSOM bằng thuật toán 3.2 [CT.9] để có được sơ đồ ở hình 3.10. Hình 3.10 là sơ đồ tối ưu hóa lập lịch theo cơ chế đường ống dựa trên kết quả lập lịch tối ưu bằng tiếp cận quy hoạch tuyến tính như sau:

- Mỗi dòng là một phép toán stream S_i, bắt đầu tại CStep tương ứng.
- Có 5 màu sắc thể hiện nhánh: *Màu xám* (Phép toán stream dùng chung), *màu xám nhạt* (Phép toán stream trung gian), *màu xanh lam* (Phép toán stream dùng cho nhánh α), *màu cam* (Phép toán stream dùng cho nhánh γ), và *màu xanh lá* (Phép toán stream dùng chung cho nhánh α và γ) (Hình 3.10).

Mô hình tối ưu hóa lập lịch theo cơ chế đường ống dựa trên quy hoạch tuyến tính (LP-based PSOM [CT.9]) dành cho xử lý stream dạng BDL thời gian thực đã tích hợp ràng buộc phụ thuộc, tái sử dụng tài nguyên, và ràng buộc không chồng lắp

vào trong một mô hình tối ưu hóa thống nhất. Mô hình này giúp giảm độ trễ, cải thiện thông lượng, cân bằng tải và tối ưu hóa việc sử dụng tài nguyên hệ thống. Luận án dựa vào *tiểu mục 1.7 (Chương 1)* để phân tích so sánh với các phương pháp hoặc các mô hình hiện có (Bảng 3.8 và bảng 3.9), các mô hình xử lý theo cửa sổ, cũng như các phương pháp học máy và học sâu đã cho thấy tính hiệu quả và khả năng mở rộng của giải pháp đề xuất đối với các bài toán xử lý stream dạng BDL thời gian thực phức tạp.

Trích xuất giá trị (Tối ưu)	$C_{\max}$	Sơ đồ tối ưu hóa lập lịch theo cơ chế đường ống
$S_1 = 1$ $S_2 = 2$ $S_3 = 1$ $S_4 = 3$ $S_5 = 3$ $S_6 = 3$ $S_7 = 4$ $S_8 = 1$ $S_9 = 4$ $S_{10} = 2$ $S_{11} = 4$ $S_{12} = 5$ $S_{13} = 5$	5	

Hình 3.10. Sơ đồ tối ưu hóa lập lịch theo cơ chế đường ống dựa vào quy hoạch tuyến tính [CT.9]

Bảng 3.8. Kết quả khảo sát những phương pháp lập lịch theo cơ đường ống

Nghiên cứu	Phương pháp	Kỹ thuật tối ưu hóa	Chỉ số cải thiện chính	Kết quả	Tính năng bổ sung
[55]	Lập lịch tuyến tính theo pipeline sử dụng phương trình Diophantine	Thuật toán Euclid mở rộng	Thông lượng, kích thước bộ đệm	Tăng thông lượng 25–40%, giảm kích thước bộ đệm tới 45%	Áp dụng cho Apache Storm
[56]	Quy hoạch tuyến tính (LP) cho lập lịch tác vụ stream phân tán	Mô hình LP cho phân bổ tài nguyên	Tránh tắc nghẽn, tối ưu tài nguyên	Không nêu định lượng cụ thể	Hệ thống phân tán nói chung
[84]	Lập lịch dựa trên LP cho phân tích stream trên Cloud	Tích hợp LP, cơ chế sao chép và giám sát thời gian thực	Chi phí thuê VM, độ trễ truy vấn, tuân thủ deadline	Giảm độ trễ 65,8%, giảm vi phạm deadline 41%	Lập lịch thích ứng cho môi trường điện toán đám mây
Mô hình quy hoạch tuyến tính đề xuất	LP-Based PSOM [CT.9]	Mô hình này: Phụ thuộc tác vụ, tái sử dụng	Tổng độ trễ, thông lượng, hiệu quả tài	Giảm độ trễ 61,54% (từ 13 xuống 5 CStep), tăng	Tinh giản CStep (2–3 phép toán mỗi bước),

Nghiên cứu	Phương pháp	Kỹ thuật tối ưu hóa	Chỉ số cải thiện chính	Kết quả	Tính năng bổ sung
của luận án		tài nguyên, ràng buộc không chồng lắp	nguyên, cân bằng tải	thông lượng, cân bằng tải	tái sử dụng thanh ghi, tập trung cho BDL

Bảng 3.9. So sánh các phương pháp/mô hình lập lịch dữ liệu stream

Phương pháp/Mô hình	Phương pháp lập lịch	Tối ưu hóa độ trễ	Tối ưu hóa thông lượng	Tối ưu hóa tài nguyên	Khả năng thích ứng tại runtime	Hỗ trợ pipeline
Mô hình quy hoạch tuyến tính đề xuất của luận án	Lập lịch pipeline dựa trên quy hoạch tuyến tính	Cao (Giảm số bước CStep)	Thực thi song	Tái sử dụng thanh ghi và dữ liệu trung gian	Tĩnh (Có thể mở rộng)	Hỗ trợ pipeline đầy đủ
Mô hình Dataflow [51]	Lập lịch dựa trên cửa sổ	Tối ưu hóa theo cửa sổ	Mở rộng tổng hợp theo cửa sổ	Quản lý trạng thái	Căn chỉnh thời gian sự kiện	Pipeline chuỗi một phần
Lập lịch học tăng cường sâu [52]	Lập lịch theo học tăng cường sâu	Dự đoán độ trễ	Cân bằng tải	Học phân bổ tài nguyên	Thích ứng rất cao	Tích hợp pipeline phức tạp
Học tăng cường sâu cho đám mây [53]	Học tăng cường sâu cho đám mây	Thích ứng thời gian thực	Điều chỉnh nhiệm vụ động	Học liên tục	Thích ứng cao	Tích hợp pipeline phức tạp
Học chủ động [54]	Học chủ động thời gian thực cho stream	Tự động học thích ứng	Cân bằng tải	Học từ chọn mẫu thông tin	Thích ứng rất cao	Pipeline hạn chế
Lập lịch theo quy hoạch tuyết tính trước [55]	Quy hoạch tuyến tính	Giảm độ trễ một phần	Thông lượng giới hạn	Hiệu quả tài nguyên biến đổi	Tập trung đám mây/Tài nguyên	Mô hình tác vụ riêng lẻ
Lập lịch điều khiển theo mô hình [56]	Lập lịch theo mô hình	Độ trễ ổn định	Tăng tốc xử lý	Dự đoán tài nguyên	Thích ứng với tốc độ dữ liệu đầu vào	Pipeline ngầm định

3.2.3. Mức chuyển đổi thanh ghi với thuật toán 3.3

Trong việc mô tả quá trình tổng hợp tính toán BDL dựa trên phép tính stream, trong đó tính toán các stream dạng BDL của một mảng các stream được ký hiệu là a và phân bổ phép toán stream tùy ý. Cụ thể là $a[0] := drop(\sigma)$, $a[1] := zip(a[0])$, $a[2] := a[1] + a[0]$, với i là số nguyên dương $i \geq 3$ và lặp cho đến khi không còn

số lượng đối số nào khả dụng, cung cấp $a[i] := a[i-1] + a[i-2] + a[i-3]$ và ánh xạ các stream đến đầu ra stream γ như được mô tả qua thuật toán 3.3 [CT.2].

Cơ chế lập lịch của tiến trình tổng hợp RTL tính toán BDL dựa trên phép tính stream đi qua bốn bước như sau: phân hoạch các phép toán stream dạng BDL, lập lịch cho tính toán BDL, cấp phát và liên kết thanh ghi, và cấp phát và liên kết đơn vị chức năng áp dụng cho thuật toán 3.3.

Thuật toán 3.3: Xử lý tính toán 01 stream đầu vào bằng các phép toán xử lý stream *(với vòng lặp)* cho kết quả đầu ra là 01 stream.

Đầu vào: $\sigma \in \mathbb{A}^{\omega}$
Đầu ra : $\gamma \in \mathbb{A}^{\omega}$
1. Begin
2. $a[0] := drop(\sigma)$;
3. $\gamma := take(a[0])$;
4. $a[1] := zip(a[0])$;
5. $\gamma := a[1]$;
6. $a[2] := a[1] + a[0]$;
7. $\gamma := a[2]$;
8. $i := 3$;
9. Loop
10. $a[i] := a[i-1] + a[i-2] + a[i-3]$;
11. $\gamma := a[i]$;
12. $i := i + 1$;
13. Exit when i>=10;
14. End loop
15. Print(γ);
16. End

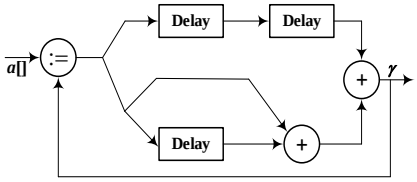
3.2.3.1. Phân hoạch các phép toán stream dạng BDL

Sơ đồ tính toán BDL dựa trên phép tính stream liên quan đến thuật toán 3.3 được mô tả trong hình 3.11. Sơ đồ này được phân hoạch ba phép toán stream sau đây: phép toán assignment $(:=)$, phép toán merging $(+)$ và phép toán registering (phép toán độ trễ đơn vị). Phép toán độ trễ đơn vị được xem xét gồm một ô nhớ một nơi lưu trữ stream dạng BDL ban đầu σ. Tại thời điểm đầu, phép toán độ trễ đơn vị bắt đầu biểu diễn của nó bằng cách phát ra stream dạng BDL ban đầu σ, trong khi đồng thời nhập stream dạng BDL tiếp theo để lưu trữ trong ô nhớ một nơi. Tại mọi thời điểm nào trong tương lai với $n \geq 1$, giá trị thứ $(n-1)^{th}$ trong ô nhớ đầu ra và stream dạng BDL mới thứ n^{th} được đưa vào và được lưu trữ.

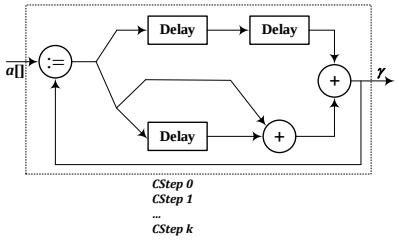
3.2.3.2. Lập lịch cho tính toán BDL

Việc lập lịch cho tính toán BDL dựa trên phép toán stream không chỉ là vì nó được xác định dựa trên tốc độ triển khai (số lượng CStep) và kích thước triển khai (tài nguyên phần cứng) được yêu cầu. Trong thuật toán 3.3, hai lịch biểu tiềm năng của các phép toán stream dạng BDL được xem xét như sau: lập lịch cho một CStep

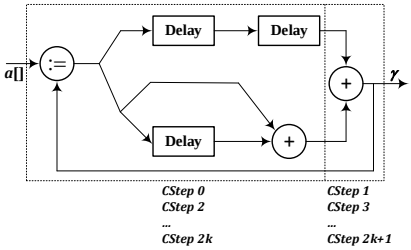
duy nhất trong hình 3.12 và lập lịch cho hai CStep trong hình 3.13.



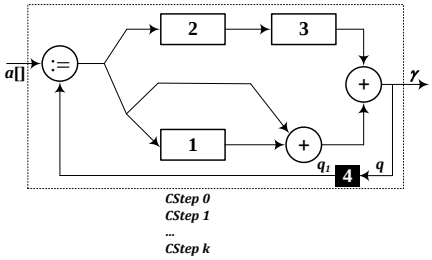
Hình 3.11. Sơ đồ tính toán BDL dựa trên phép toán stream



Hình 3.12. Sơ đồ biểu diễn lập lịch cho một CStep duy nhất



Hình 3.13. Sơ đồ biểu diễn lập lịch cho hai CStep



Hình 3.14. Sơ đồ cấp phát và liên kết thanh ghi cho việc lập lịch một CStep

3.2.3.3. Cấp phát và liên kết thanh ghi

Nếu việc lập lịch cho một CStep duy nhất được sử dụng, nó đòi hỏi một thanh ghi bổ sung được đặt tên có nhãn là 4 để cấp phát và liên kết ở cuối mỗi CStep. Hơn nữa, ba độ trễ đơn vị được thay thế bằng ba thanh ghi được đặt tên là 1, 2, 3 như trong hình 3.14. Do đó, phương trình vi phân hành vi xác định hành vi của thuật toán 3.3 ở bảng 3.10 như sau:

Bảng 3.10. Phương trình vi phân hành vi xác định hành vi thuật toán 3.3

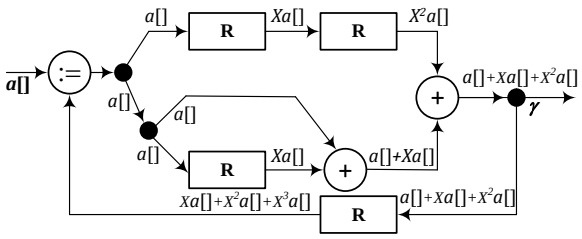
Phương trình vi phân hành vi	Giá trị ban đầu
$a^{(3)}[] = a^{(2)}[] + a^{(1)}[] + a[]$	$a[0] = drop(\sigma)$, $a^{(1)}[0] = zip(a[0])$ $a^{(2)}[0] = a^{(1)}[0] + a[0]$

Để giải phương trình vi phân hành vi này: $a^{(3)}[] = a^{(2)}[] + a^{(1)}[] + a[]$, định lý 2.1 được sử dụng để thu được các phương trình vi phân như sau: $a[] = 1 + X \times a^{(1)}[]$, $a^{(1)}[] = 1 + X \times a^{(2)}[]$, $a^{(2)}[] = 2 + X \times a^{(3)}[]$.

Những phương trình này ám chỉ ra rằng $a^{(1)}[] = 1 + 2X + X^2 \times a^{(3)}[]$, $a[] = 1 + X + 2X^2 + X^3 \times a^{(3)}[]$ và $a^{(3)}[] = a^{(2)}[] + a^{(1)}[] + a[] = \left(2 + X \times a^{(3)}[]\right) + \left(1 + 2X + X^2 \times a^{(3)}[]\right) + \left(1 + X + 2X^2 + X^3 \times a^{(3)}[]\right)$.

Do đó, $a^{(3)}[] = (4 + 3X + 2X^2) \times Rev_3(1 - X - X^2 - X^3)$ và $a[] = 1 + X + 2X^2 + X^3 \times \left((4 + 3X + 2X^2) \times Rev_3(1 - X - X^2 - X^3)\right) = 1 + X + 2X^2 + \left((4X^3 + 3X^4 + 2X^5) \times Rev_3(1 - X - X^2 - X^3)\right)$. Điều này mang lại kết quả theo phương trình vi phân sau: $a[] = 1 + X + 2X^2 + \left((4X^3 + 3X^4 + 2X^5) \times Rev_3(1 - X - X^2 - X^3)\right)$.

Sơ đồ stream trong hình 3.15 cũng được xem như là một triển khai của thuật toán 3.3 và hàm hành vi xác định hành vi của stream dạng BDL đầu ra γ ở bảng 3.11.



Hình 3.15. Sơ đồ tính toán BDL trên phép tính stream cho việc lập lịch một CStep

Bảng 3.11. Phương trình vi phân hành vi xác định hành vi của stream đầu ra γ

Phương trình vi phân hành vi	Giá trị ban đầu
$\gamma = a[] + X \times a[] + X^2 \times a[]$	$a[0] = drop(\sigma)$, $a^{(1)}[0] = zip(a[0])$ $a^{(2)}[0] = a^{(1)}[0] + a[0]$

Mệnh đề 3.2 (Mô phỏng hai chiều): Hai stream dạng BDL đầu ra $a[] + X \times a[] + X^2 \times a[]$ và $X \times a[] + X^2 \times a[] + X^3 \times a[]$ là tương đương theo mô phỏng hai chiều.

Chứng minh: Theo mệnh đề 3.1, tồn tại quan hệ mô phỏng hai chiều $B \subseteq \mathbb{A}^\omega \times \mathbb{A}^\omega$ với $\langle a[] + X \times a[] + X^2 \times a[], X \times a[] + X^2 \times a[] + X^3 \times a[] \rangle \in B$. Quan hệ B được xây dựng qua các giai đoạn, bằng cách tính đạo hàm của cả hai stream $\alpha_1 = a[] + X \times a[] + X^2 \times a[]$ và $\alpha_2 = X \times a[] + X^2 \times a[] + X^3 \times a[]$ theo từng bước, trong đó α_1 và α_2 là hai stream con của stream α. Các cặp này được xét trong B là $\{\langle \alpha_1^{i+k}, \alpha_2^{i+k} \rangle \mid 0 \le i \le n, k \ge 0\}$. □

Hệ quả 3.2 (Đồng quy nạp): Nếu $(a[] + X \times a[] + X^2 \times a[]) \sim (X \times a[] + X^2 \times a[] + X^3 \times a[])$, thì $(a[] + X \times a[] + X^2 \times a[]) = (X \times a[] + X^2 \times a[] + X^3 \times a[])$.

Chứng minh: Dựa trên hệ quả 3.1, xét hai stream dạng BDL đầu ra $\alpha_1 = a[] + X \times a[] + X^2 \times a[]$ và $\alpha_2 = X \times a[] + X^2 \times a[] + X^3 \times a[]$. Theo *mệnh đề 3.2*, quan hệ $B \subseteq \mathbb{A}^\omega \times \mathbb{A}^\omega$ chứa cặp $\langle \alpha_1^i, \alpha_2^{i+k} \rangle$. Dùng quy nạp trên k, với $0 \le i \le n$, $k \ge 0$, ta có $\langle \alpha_1^i, \alpha_2^{i+k} \rangle \in B$. Do đó, $\alpha_1^i(0) = \alpha_2^{i+k}(0)$, chứng minh $\alpha_1 = \alpha_2$. □

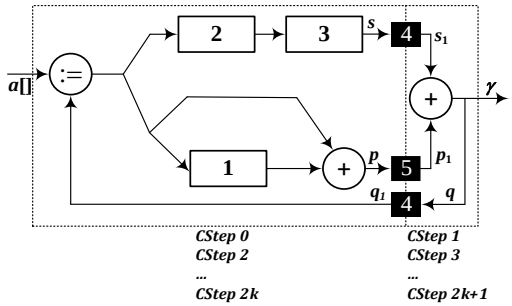
Nếu lập lịch đôi CStep được sử dụng, ba đơn vị độ trễ được thay thế ngữ nghĩa bằng ba thanh ghi được đặt tên là 1, 2, 3. Hơn nữa, hai thanh ghi bổ sung được đặt tên là 4 và 5 được cấp phát và liên kết ở cuối mỗi CStep được đánh số bằng số chẵn. Thanh ghi thứ 4 được tái sử dụng ở cuối mỗi CStep được đánh số bằng một số lẻ như trong *hình 3.16*. Do đó, phương trình vi phân xác định hành vi của thuật toán 3.3 ở bảng 3.12.

Bảng 3.12. Phương trình vi phân hành vi xác định hành vi thuật toán 3.3

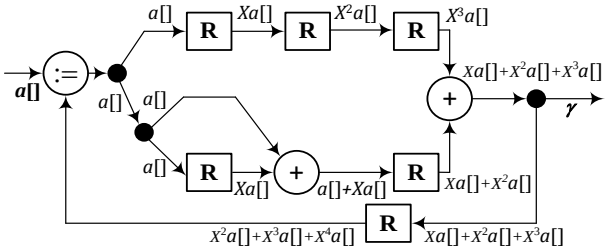
Phương trình vi phân hành vi	Giá trị ban đầu
$a^{(3)}[] = X \times \left(a^{(2)}[] + a^{(1)}[] + a[]\right)$	$a[0] = drop(\sigma)$, $a^{(1)}[0] = zip(a[0])$ $a^{(2)}[0] = a^{(1)}[0] + a[0]$

Để tìm nghiệm cho phương trình vi phân hành vi này $a^{(3)}[] = X \times \left(a^{(2)}[] + a^{(1)}[] + a[]\right)$, *định lý 2.1* được áp dụng như trong trường hợp một CStep duy nhất ở trên để thu được phương trình vi phân như sau: $a^{(3)}[] = X \times \left(a^{(2)}[] + a^{(1)}[] + a[]\right) =$

$X \times \left(\left(2 + X \times a^{(3)}[]\right) + \left(1 + 2X + X^2 \times a^{(3)}[]\right)\right) + \left(1 + X + 2X^2 + X^3 \times a^{(3)}[]\right)$. Do đó, $a^{(3)}[] = 4X + 3X^2 + 2X^3 \times Rev_4(1 - X^2 - X^3 - X^4)$ và $a[] = 1 + X + 2X^2 + X^3 \times \left(4X + 3X^2 + 2X^3 \times Rev_4(1 - X^2 - X^3 - X^4)\right) = 1 + X + 2X^2 + \left(4X^4 + 3X^5 + 2X^6 \times Rev_4(1 - X^2 - X^3 - X^4)\right)$. Kết quả dưới dạng $a[] = 1 + X + 2X^2 + \left(4X^4 + 3X^5 + 2X^6 \times Rev_4(1 - X^2 - X^3 - X^4)\right)$.



Hình 3.16. Sơ đồ cấp phát và liên kết thanh ghi cho lập lịch đôi CStep



Hình 3.17. Sơ đồ tính toán BDL trên phép tính stream biểu diễn lập lịch đôi CStep

Bảng 3.13. Phương trình vi phân hành vi xác định hành vi stream đầu ra γ

Phương trình vi phân hành vi	Giá trị ban đầu
$\gamma = X \times a[] + X^2 \times a[] + X^3 \times a[]$	$a[0] = \sigma, a^{(1)}[0] = a[0]$ $a^{(2)}[0] = a^{(1)}[0] + a[0]$

Sơ đồ stream biểu diễn trong *hình 3.17*, cũng được xem như là một triển khai của thuật toán 3.3 và hàm hành vi xác định hành vi stream đầu ra γ dạng BDL ở bảng 3.13.

Thay thế $a[] = 1 + X + 2X^2 + \left(4X^4 + 3X^5 + 2X^6 \times Rev_4(1 - X^2 - X^3 - X^4)\right)$ cho phương trình vi phân này sẽ cung cấp hành vi stream đầu ra γ dưới dạng đóng: $\gamma = 1 + X + 2X^2 + \left(4X^4 + 3X^5 + 2X^6 \times Rev_4(1 - X^2 - X^3 - X^4)\right) \times (X + X^2 + X^3)$.

Mệnh đề 3.3 (Mô phỏng hai chiều): Hai stream dạng BDL đầu ra: $X \times a[] + X^2 \times a[] + X^3 \times a[]$ và $X^2 \times a[] + X^3 \times a[] + X^4 \times a[]$ là tương đương theo mô phỏng hai chiều.

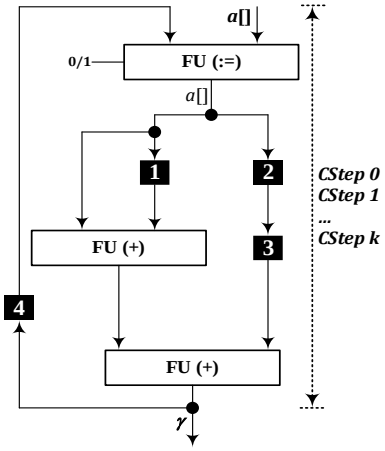
Chứng minh: Dựa trên mệnh đề 3.2, tồn tại quan hệ mô phỏng hai chiều $B \subseteq \mathbb{A}^\omega \times \mathbb{A}^\omega$ với $\langle X \times a[] + X^2 \times a[] + X^3 \times a[], X^2 \times a[] + X^3 \times a[] + X^4 \times a[]\rangle \in B$. Quan hệ B được xây dựng bằng cách tính đạo hàm của cả hai stream $\alpha_1 = X \times a[] + X^2 \times a[] + X^3 \times a[]$ và $\alpha_2 = X^2 \times a[] + X^3 \times a[] + X^4 \times a[]$ theo từng bước, trong đó α_1 và α_2 là hai stream con của α. Các cặp được tạo ra trong B là các cặp: $\{\langle \alpha_1^{i+k}, \alpha_2^{i+k}\rangle \mid 0 \le i \le n, k \ge 0\}$. □

Hệ quả 3.3 (Đồng quy nạp): Nếu $(X \times a[] + X^2 \times a[] + X^3 \times a[]) \sim (X^2 \times a[] + X^3 \times a[] + X^4 \times a[])$, thì $(X \times a[] + X^2 \times a[] + X^3 \times a[]) = (X^2 \times a[] + X^3 \times a[] + X^4 \times a[])$.

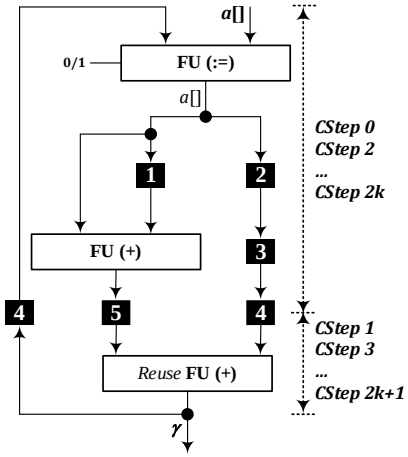
Chứng minh: Theo *hệ quả 3.2*, xét hai stream đầu ra $\alpha_1 = X \times a[] + X^2 \times a[] + X^3 \times a[]$ và $X^2 \times a[] + X^3 \times a[] + X^4 \times a[]$. *Mệnh đề 3.3* chỉ ra rằng

quan hệ $B \subseteq \mathbb{A}^\omega \times \mathbb{A}^\omega$ chứa cặp $\langle \alpha_1^i, \alpha_2^{i+k} \rangle$. Bằng quy nạp trên k, với $0 \leq i \leq n, k \geq 0$, ta có cặp $\langle \alpha_1^i, \alpha_2^{i+k} \rangle \in B$. Do đó, $\alpha_1^i(0) = \alpha_2^{i+k}(0)$, chứng minh $\alpha_1 = \alpha_2$. □

3.2.3.4. Cấp phát và liên kết đơn vị chức năng



Hình 3.18. Sơ đồ cấp phát và liên kết đơn vị chức năng cho việc lập lịch CStep đơn



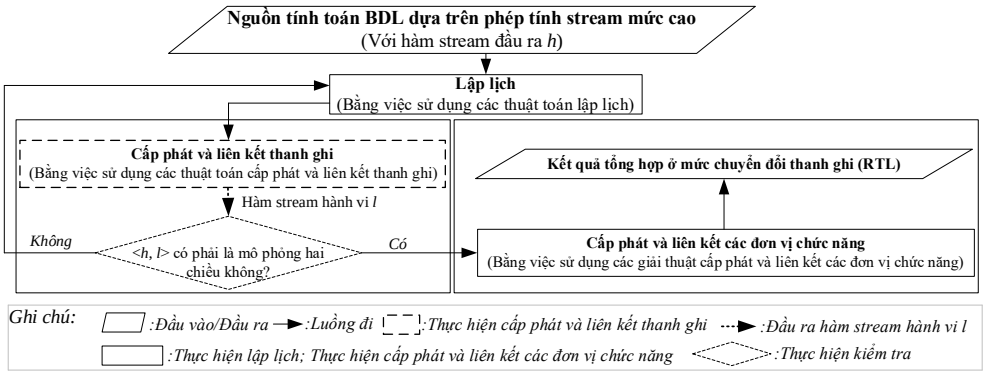
Hình 3.19. Sơ đồ cấp phát và liên kết đơn vị chức năng xử lý cho lập lịch CStep đôi

Nếu sử dụng lập lịch CStep đơn, thì hai đơn vị chức năng riêng biệt được triển khai cho hai phép toán merging $(+)$ và một phép toán assignment $(:=)$, với tín hiệu điều khiển $0/1$ chọn triển khai phù hợp (*Hình 3.18*). Nếu tín hiệu điều khiển được chuyển đổi thành 0, thì stream đầu vào của $a[]$ đi đến đầu ra ban đầu, ngược lại $a[]$ sẽ được gán bởi stream đầu vào thứ hai trước khi được xuất ra. *Hình 3.18* là một sơ đồ thể hiện cách các đơn vị chức năng xử lý được cấp phát và liên kết trong quá trình lập lịch CStep đơn của thuật toán 3.3.

Nếu sử dụng lập lịch CStep đôi cho stream dạng BDL, chỉ cần hai đơn vị chức năng xử lý. Trong mỗi CStep được đánh số bằng các số chẵn, một đơn vị chức năng được sử dụng để thực hiện phép toán merging $(+)$ và một đơn vị chức năng khác được sử dụng để thực hiện phép toán assignment $(:=)$. Mỗi CStep được đánh số các số lẻ, đơn vị chức năng đó để trộn được sử dụng lại như được thể hiện ở *hình 3.19*.

3.3. Tiếp cận đồng quy nạp để kiểm chứng việc tổng hợp tính toán BDL

3.3.1. Kiểm chứng tổng hợp



Hình 3.20. Lưu đồ của phương pháp tiếp cận đồng quy nạp trong việc kiểm chứng quá trình tổng hợp của tính toán BDL dựa trên phép tính stream

Quá trình tổng hợp tính toán BDL dựa trên phép tính stream được kiểm chứng sau khi chạy xong là một kỹ thuật kiểm chứng thực tế nhằm mục đích tích hợp vào các bước tổng hợp. Ngoài ra, với các kỹ thuật kiểm chứng sau quá trình tổng hợp hiện tại, việc kiểm chứng quá trình tổng hợp tính toán BDL dựa trên phép tính stream có thể được thực hiện để đảm bảo tính chính xác trong quá trình tổng hợp, hình 3.20 cho tiếp cận đồng quy nạp. Như vậy, với việc kiểm chứng này, phương pháp tiếp cận đồng quy nạp đảm bảo tính chính xác của quá trình tổng hợp.

3.3.2. Tiếp cận đồng quy nạp để kiểm chứng tổng hợp tính toán BDL

Sự khác biệt so với việc kiểm chứng sau quá trình tổng hợp là thông tin về bước tổng hợp được thực hiện và được sử dụng có lợi thế trong quá trình kiểm chứng. Tuy nhiên, thông tin về cách thức tổng hợp được thực hiện vẫn chưa biết. Do đó, phương pháp tiếp cận đồng quy nạp trong việc kiểm chứng quá trình tổng hợp tính toán BDL dựa trên phép tính stream không bị ràng buộc bởi tiến trình thuật toán được thực tế hóa trong tiến trình lập lịch.

Sử dụng mối quan hệ mô phỏng hai chiều và hàm stream, kỹ thuật kiểm chứng được tích hợp vào bước cấp phát và liên kết thanh ghi của quá trình tổng hợp (gồm bốn bước *ở tiểu mục 3.2.3*). Với phương pháp này, mối quan hệ mô phỏng hai chiều được xem xét giữa hàm stream đầu ra h biểu diễn đầu ra từ nguồn tính toán BDL dựa trên phép tính stream ở mức cao và hàm stream hành vi l biểu diễn đầu ra sau cấp phát và liên kết thanh ghi. Minh họa việc xét các mối quan hệ mô phỏng hai chiều trong các *mệnh đề 3.1 - 3.3*, dưới đây là mô phỏng hai chiều ở bảng 3.14.

Kỹ thuật kiểm chứng tổng hợp tính toán BDL dựa trên phép tính stream khắc phục hạn chế về thời gian của kiểm chứng sau tổng hợp và là lựa chọn thay thế hiệu quả cho các phương pháp hiện có.

Bảng 3.14. Mô phỏng hai chiều giữa hàm stream đầu ra h và hàm stream hành vi l

h		l
σ (hoặc τ)	~	$X^n \times \mu$ (hoặc $X^n \times \delta$), với $n \geq 0$
$a[] + X \times a[] + X^2 \times a[]$		$X \times a[] + X^2 \times a[] + X^3 \times a[]$
$X \times a[] + X^2 \times a[] + X^3 \times a[]$		$X^2 \times a[] + X^3 \times a[] + X^4 \times a[]$

Suy ra, mối quan hệ mô phỏng hai chiều tổng quát được xem xét giữa hàm stream đầu ra h và hàm stream hành vi l như bảng 3.15 sau đây:

Bảng 3.15. Mô phỏng hai chiều tổng quát giữa h và l

h với $m \in \mathbb{N}$	~	l với $n \in \mathbb{N}$
$X^m \times a[] + X^{m+1} \times a[] + X^{m+2} \times a[]$		$X^n \times a[] + X^{n+1} \times a[] + X^{n+2} \times a[]$

Như vậy, chương 3 này cũng đã giới thiệu thêm kỹ thuật kiểm chứng ngược (đồng quy nạp) cho quá trình tổng hợp tính toán BDL dựa trên phép tính stream, không phụ thuộc vào thuật toán dùng trong cấp phát và liên kết thanh ghi. Kỹ thuật

này sử dụng nguyên lý chứng minh ngược và phép tính stream để đặc tả, cho thấy kiểm chứng hình thức dựa trên mô phỏng hai chiều và hàm stream hành vi là giải pháp thay thế tiềm năng cho kiểm chứng truyền thống sau tổng hợp.

3.4. Phần kết chương 3

Chương 3 đã xây dựng cơ chế lập lịch để xử lý stream dạng BDL. Lập lịch này cho phép đường ống hoạt động ở đó tất cả các tài nguyên phần cứng gồm các thanh ghi cùng với các đơn vị chức năng được tái sử dụng trong suốt các bước điều khiển khác nhau nhằm giảm độ trễ, tăng thông lượng, cân bằng tải và tối ưu sử dụng tài nguyên hệ thống. Luận án áp dụng nguyên lý tiếp cận đồng quy nạp để nghiên cứu kiểm chứng tổng hợp tính toán BDL dựa vào phép tính stream. Qua kết quả nghiên cứu của chương 3 này giúp cho tác giả hình dung ra các công việc tiếp theo được thực hiện ở chương 4.

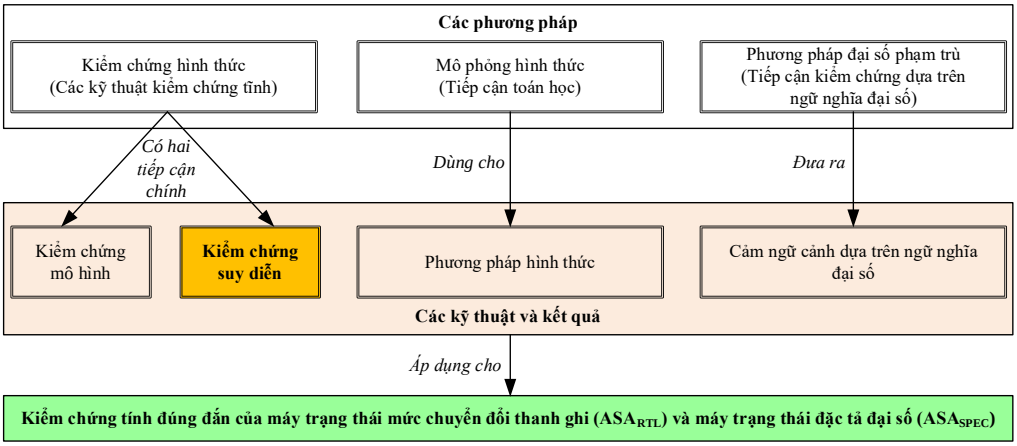
Chương 4 nghiên cứu xây dựng mô hình ngữ nghĩa đại số ASM để xử lý stream dạng BDL. Mô hình này cho phép đặc tả mối quan hệ giữa máy trạng thái mức chuyển đổi thanh ghi (ASA_{RTL}) và máy trạng thái đặc tả đại số (ASA_{SPEC}) của thuật toán 4.2 và 4.4 để kiểm chứng kết quả tổng hợp RTL và SPEC.

Chương 4. NGỮ NGHĨA ĐẠI SỐ MỨC CHUYỂN ĐỔI THANH GHI TRONG TỔNG HỢP TÍNH TOÁN BDL DỰA TRÊN PHÉP TÍNH STREAM

Chương này tập trung trình bày đóng góp tạo ra mô hình ngữ nghĩa đại số (ASM) để xử lý stream dạng BDL được công bố trong công trình [CT.4] và [CT.6]. Mười phép toán xử lý stream trong công trình [CT.2] được sử dụng tạo mô hình này cho phép đặc tả mối quan hệ giữa máy trạng thái mức chuyển đổi thanh ghi (ASA_{RTL}) và máy trạng thái đặc tả đại số (ASA_{SPEC}) của thuật toán 4.2 và thuật toán 4.4 để kiểm chứng các kết quả tổng hợp RTL và SPEC giúp luận án có được một trong ba kết luận sau: Một kết quả tổng hợp RTL và SPEC dựa trên ngữ nghĩa đại số là chính xác nếu và chỉ nếu $\mathcal{C}(\text{RTL}) = \mathcal{C}(\text{SPEC})$; Nếu $\mathcal{C}(\text{RTL}) \subset \mathcal{C}(\text{SPEC})$ thì tồn tại một tính toán $\mathcal{T} \in \mathcal{C}(\text{SPEC}) \setminus \mathcal{C}(\text{RTL})$ không được tổng hợp trong kết quả RTL dẫn đến kết quả của tổng hợp RTL không chính xác; Nếu $\mathcal{C}(\text{SPEC}) \subset \mathcal{C}(\text{RTL})$ thì đặc tả và tổng hợp RTL không tương đương suy ra hành vi tính toán của chúng là khác nhau.

4.1. Giới thiệu

Kiểm chứng tính đúng đắn trong tính toán stream dạng BDL dựa trên các phép toán stream khi phân tích và xử lý dữ liệu lớn livestream là rất phức tạp. Trong quá trình tổng hợp tính toán BDL dựa trên các phép toán stream [CT.2][85], việc kiểm chứng tính đúng đắn của máy trạng thái mức chuyển đổi thanh ghi (ASA_{RTL}) và máy trạng thái đặc tả đại số (ASA_{SPEC}) dựa trên ngữ nghĩa đại số là nhiệm vụ khó khăn. Có ba phương pháp kiểm chứng như sau: *Kiểm chứng hình thức*, *mô phỏng hình thức*, và *phương pháp đại số phạm trù* [CT.4][CT.6][86-89] (Hình 4.1).



Hình 4.1. Kiểm chứng tính đúng đắn các đặc tả thuật toán và máy trạng thái RTL

Kiểm chứng hình thức là một kỹ thuật được sử dụng để kiểm tra tính đúng đắn của các hệ thống kỹ thuật phần mềm đối với mô tả hình thức [90] và được sử dụng để kiểm chứng chức năng của các thiết kế RTL [91-93]. Kiểm chứng hình thức sử dụng phân tích tĩnh dựa trên các biến đổi toán học để định nghĩa tính đúng đắn của

hành vi phần mềm [94], đặc biệt là những hệ thống được biểu đạt bằng ngôn ngữ lập trình mức cao. Phương pháp này được áp dụng vào kiểm chứng sự tương đương chức năng giữa ASA_{SPEC} và ASA_{RTL}. Hai tiếp cận chính của kiểm chứng hình thức trong kỹ thuật kiểm chứng tĩnh là:

- *Kiểm chứng mô hình:* Liên quan đến việc khám phá toàn diện mô hình toán học của hệ thống một cách hệ thống. Đây là một dạng kiểm chứng sử dụng để kiểm tra xem một mô hình có đáp ứng một đặc tả cụ thể hay không [95].
- *Kiểm chứng suy diễn:* Liên quan đến việc tạo ra bộ các nhiệm vụ chứng minh toán học từ hệ thống [96]. Tính đúng đắn của bộ các nhiệm vụ này ngụ ý hệ thống tuân theo đặc tả của nó. Các nhiệm vụ này sau đó được giải quyết bằng cách sử dụng các bộ chứng minh định lý tự động.

Mô phỏng hình thức là một kỹ thuật được sử dụng trong lĩnh vực kỹ thuật phần mềm để đảm bảo rằng một thiết kế có thể đáp ứng các yêu cầu mô tả của nó. Đây là một tiếp cận toán học sử dụng các phương pháp hình thức để kiểm chứng tính đúng đắn của một thiết kế.

Phương pháp đại số phạm trù là một tiếp cận kiểm chứng dựa trên ngữ nghĩa đại số, được sử dụng trong nghiên cứu để xây dựng một nền tảng hình thức cho mô hình cảm ngữ cảnh để đạt được cảm ngữ cảnh dựa trên ngữ nghĩa đại số.

Luận án đề xuất một tiếp cận đặc tả mối quan hệ giữa ASA_{RTL} và ASA_{SPEC} dựa trên ngữ nghĩa đại số trong quá trình tổng hợp tính toán BDL dựa trên phép tính stream. Mục tiêu của chương này là xây dựng mô hình ngữ nghĩa đại số làm cơ sở để phân tích và kiểm chứng mối quan hệ giữa ASA_{RTL} và ASA_{SPEC} dựa vào ngữ nghĩa đại số trong quá trình tổng hợp tính toán stream dạng BDL nhằm đáp ứng một trong ba kết luận hữu ích cho kiểm chứng tính đúng đắn của ASA_{RTL} và ASA_{SPEC}.

4.2. Khái niệm không gian Chu

4.2.1. Không gian Chu

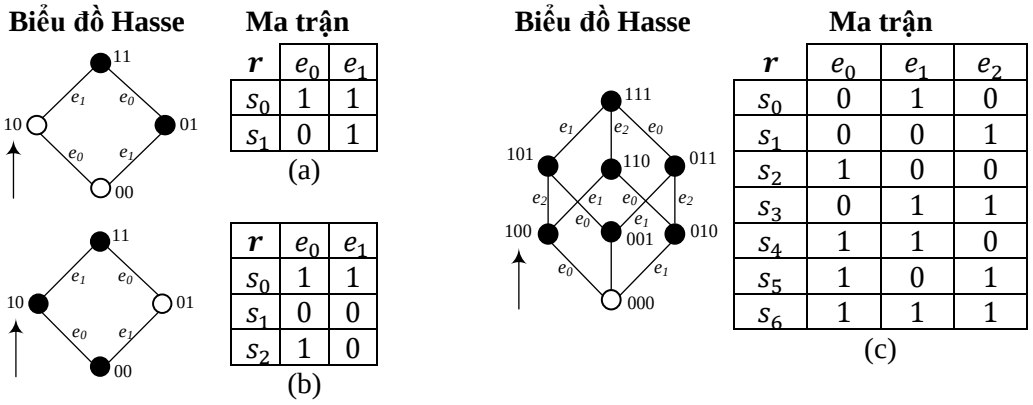
Dựa vào *Định nghĩa 1.5* (*Tiểu mục 1.6.11*, *Chương 1*), hàm r có thể được biểu diễn dưới dạng một ma trận với các hàng là các trạng thái trong X, các cột là các sự kiện trong A, và các giá trị tại mỗi ô là $r(s,e) \in K$ biểu thị mức độ mà trạng thái s thỏa mãn sự kiện e (hình 4.2).

$\boldsymbol{r}$	e_0	e_1	e_2
s_0	0	1	0
s_1	1	0	1
s_2	0	0	1

Hình 4.2. Một không gian Chu trên tập giá trị {0,1} thông qua các mối quan hệ $\{s_0,s_1,s_2\}$ và $\{e_0,e_1,e_2\}$

Với A và X được xem như hai tập toán học chung. Hình 4.3 đưa ra một số không gian Chu cùng với các minh họa về biểu đồ Hasse. Các phần tử của sự kiện A được

biểu diễn bởi $e_0, e_1, e_2, e_3, \dots e_n$, với $0 \le i \le n$ và các phần tử của trạng thái X được biểu diễn bởi $s_0, s_1, s_2, s_3, \dots s_m$, với $0 \le j \le m$.



r	e_0	e_1
s_0	1	1
s_1	0	1

r	e_0	e_1
s_0	1	1
s_1	0	0
s_2	1	0

r	e_0	e_1	e_2
s_0	0	1	0
s_1	0	0	1
s_2	1	0	0
s_3	0	1	1
s_4	1	1	0
s_5	1	0	1
s_6	1	1	1

Hình 4.3. Các không gian Chu và những mô tả của nó bằng ma trận (a), (b) và (c)

Trong các nghiên cứu [97], họ đã biểu diễn A như là tập sự kiện và X như là tập trạng thái. Một trạng thái được định nghĩa bởi một mối quan hệ xuất hiện $r(e, s)$ là đúng mà nơi sự kiện e đã xảy ra trong trạng thái s. Do đó, mỗi trạng thái s là một tập con của A chứa các sự kiện đã xảy ra trong trạng thái s. Ngữ nghĩa đại số được xem như một không gian Chu C được đặc tả bởi bộ ba (A, X, r), trong đó $A = \{e_0, e_1, e_2, e_3, \dots, e_n\}$ là tập sự kiện với $1 \le i \le n$, $X = \{s_0, s_1, s_2, s_3, \dots, s_n\}$ là tập trạng thái với $1 \le i \le n$, và $r: A \times X \to \{0, 1\}$ biểu diễn cho mối quan hệ xuất hiện, được hiểu là $r(e, s) = 1$, nếu sự kiện e đã xảy ra trong trạng thái s và $r(e, s) = 0$, ngược lại. Đối với mọi $e \in A$, mỗi trạng thái $s_i \in X = 2^A$ được định nghĩa theo r bởi $s_i = \{e \mid r(e, s_i) = 1\}$.

t	e_3	e_4
s_3	0	1
s_4	1	0

Hình 4.4. Các không gian Chu được trình bày qua mối quan hệ t của tập hợp trạng thái $\{s_3, s_4\}$ và tập hợp sự kiện $\{e_3, e_4\}$

Không gian Chu có thể được viết dưới dạng $(\{s_0, s_1, s_2\}, r, \{e_0, e_1, e_2\})$, trong đó r được định nghĩa bởi các ma trận ở trên. Các giá trị s_0, s_1 và s_2 có thể được xem như là các trạng thái có các giá trị khác nhau cho các sự kiện e_0, e_1 và e_2 tương ứng. Minh họa trong hình 4.2, s_0 có giá trị là 0 cho e_0, 1 cho e_1, và 0 cho e_2, tương ứng. Một biến đổi Chu là một cặp hàm (f, g) ánh xạ một không gian Chu sang một không gian Chu khác, sao cho các giá trị của các sự kiện phải được bảo tồn. Minh họa nếu có một không gian Chu khác $(\{s_3, s_4\}, t, \{e_3, e_4\})$, ở đó mối quan hệ t được định nghĩa bởi ma trận ở hình 4.4.

Sau đó, một biến đổi Chu có thể có từ $(\{s_0, s_1, s_2\}, r, \{e_0, e_1, e_2\})$ đến $(\{s_3, s_4\}, t, \{e_3, e_4\})$ được cho bởi một cặp hàm (f, g), trong đó hàm f được định nghĩa bằng ma trận ở hình 4.5 và hàm g được định nghĩa bằng ma trận ở hình 4.6.

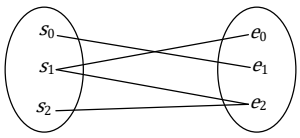
s_0	s_1	s_2
s_3	s_4	s_3

Hình 4.5. Các biến đổi Chu từ r đến t được cho bởi hàm f

e_3	e_4
e_1	e_2

Hình 4.6. Các biến đổi Chu từ r đến t được cho bởi hàm g

Điều này có nghĩa là hàm f ánh xạ s_0 và s_2 thành s_3, và s_1 thành s_4, trong khi g ánh xạ e_3 thành e_1 và e_4 thành e_2. Điều kiện kề cho các biến đổi không gian Chu được thỏa mãn, vì $t(f(s_0), e_3) = r\big(s_0, g(e_3)\big)$ cho tất cả s_0 và e_3, và tương tự cho e_4. Minh họa $t(f(s_0), e_3) = t(s_3, e_3) = t(s_3, e_1) = r(s_0, e_1) = r\big(s_0, g(e_3)\big)$. Một không gian Chu cũng có thể được biểu diễn như một sơ đồ hai phía (Trong không gian Chu, sơ đồ hai phía thường được sử dụng để biểu diễn mối quan hệ giữa tập hợp các trạng thái X và tập hợp các sự kiện A), trong đó các sự kiện và các trạng thái là hai tập hợp các đỉnh và các cạnh được đánh dấu bằng các giá trị từ $K = A \times X \rightarrow \{0, 1\}$. Minh họa không gian Chu $(\{s_0, s_1, s_2\}, r, \{e_0, e_1, e_2\})$ có thể được biểu diễn bằng sơ đồ hai phía, kết hợp các trạng thái và các sự kiện trong hình 4.2 để biểu diễn sơ đồ hai phía ở hình 4.7.



Hình 4.7. Sơ đồ hai phía của các tập hợp các trạng thái $\{s_0,s_1,s_2\}$ và các sự kiện $\{e_0,e_1,e_2\}$ và mối quan hệ r liên quan đến hình 4.2.

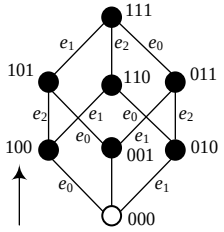
4.2.2. Ngữ nghĩa đại số như không gian Chu

Trong ngữ cảnh của ngữ nghĩa đại số, một không gian Chu C bao gồm bộ ba (A, r, X) với mối quan hệ r, trong đó $A = \{e_0, e_1, e_2, e_3, \ldots, e_n\}$, với $1 \leq i \leq n$ là tập sự kiện, $X = \{s_0, s_1, s_2, s_3, \ldots, s_m\}$, với $1 \leq j \leq m$ là tập trạng thái, và $r: A \times X \rightarrow K = \{0, 1\}$ biểu diễn cho quan hệ xảy ra được hiểu là $r(e, s) = 1$ nếu sự kiện e đã xảy ra trong trạng thái s và $r(e, s) = 0$, ngược lại là không. Đối với mọi $e \in A$, mỗi trạng thái $s_j \in 2^A$ được định nghĩa dưới dạng quan hệ r bởi $s_j = \{e \mid r(e, s_j) = 1\}$ được thể hiện chi tiết ở hình 4.8.

Ngữ nghĩa đại số trong hình 4.8 có ba sự kiện đó là $A = \{e_0, e_1, e_2\}$ và bảy trạng thái đó là $X = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6\}$, biểu diễn cho một hệ thống, ở đó bất kỳ sự kiện nào thuộc $\{e_0, e_1, e_2\}$ xảy ra và theo dữ liệu được tạo ra bởi sự kiện đó một trong hai sự kiện còn lại sẽ xảy ra. Minh họa trong trạng thái s_1, sự kiện e_2 đã xảy ra; s_3 mô tả trạng thái ở đó sự kiện e_1 đã xảy ra sau e_2 hoặc e_2 đã xảy ra sau e_1.

Một ngữ nghĩa đại số có thể được biểu diễn dưới dạng ma trận chẳng hạn như dưới dạng biểu đồ Hasse hoặc dưới dạng công thức logic [49]. Trong đặc tả ma trận,

mỗi cặp giá trị (e, s) đại diện cho giá trị của quan hệ xảy ra. Do đó, các dòng của ma trận liên quan đến các trạng thái của ngữ nghĩa đại số và các cột của ma trận liên quan đến các sự kiện của ngữ nghĩa đại số. Trong biểu đồ Hasse, một ngữ nghĩa đại số được biểu diễn bởi thứ tự bộ phận hiện tồn tại giữa các trạng thái. Trong công thức logic, ma trận được xem như bảng chân trị, đặc tả logic $f_{\mathcal{D}}$ của ngữ nghĩa đại số $\mathcal{D}$ được định nghĩa bởi hàm đặc tả luận lý: $f_{\mathcal{D}} = \bigvee_{0\le i\le n \text{ và } 0\le j\le l}\left(\wedge\left\{e_j \mid r\left(e_j, s_i\right) = 1\right\}\right) \wedge \left\{\wedge\, \bar{e}_j \mid r\left(\bar{e}_j, s_i\right) = 0\right\}$, ở đó $n \ge 0$ có dạng $n = |X| = |\{s_0, s_1, s_2, s_3, \dots, s_6\}|$ và $l = |A| = |\{e_0, e_1, e_2\}|$, và $e_j \in A$ và $\bar{e}_j$ là phần bù của e_j.



Biểu đồ Hasse

Ma trận

r	e_0	e_1	e_2	
s_0	0	1	0	$\{e_1\}$
s_1	0	0	1	$\{e_2\}$
s_2	1	0	0	$\{e_0\}$
s_3	0	1	1	$\{e_1, e_2\}$
s_4	1	1	0	$\{e_0, e_1\}$
s_5	1	0	1	$\{e_0, e_2\}$
s_6	1	1	1	$\{e_0, e_1, e_2\}$

Công thức luận lý

f_{ASM}
$\{\bar{e_0}, e_1, \bar{e_2}\}$
$\{\bar{e_0}, \bar{e_1}, e_2\}$
$\{e_0, \bar{e_1}, \bar{e_2}\}$
$\{\bar{e_0}, e_1, e_2\}$
$\{e_0, e_1, \bar{e_2}\}$
$\{e_0, \bar{e_1}, e_2\}$
$\{e_0, e_1, e_2\}$

Hình 4.8. Ngữ nghĩa đại số và những mô tả của nó như không gian Chu

Trong hình 4.8, các sự kiện $e \in A$ được xem như là các biến, các trạng thái $s \in X$ được xem như là các công thức luận lý, và mối quan hệ r xác định liệu biến có được phủ định $r(e, s_i) = 0$ hay không $r(e, s_i) = 1$. Công thức logic f là đúng cho mỗi trạng thái $s \in X$ được phép trong ngữ nghĩa đại số. Từ đó, một mô hình dựa trên ngữ nghĩa đại số (ASM) thu được có dạng $\text{ASM} = (A, r, X)$, $A = \{e_0, e_1, e_2\}$ chứa tập hợp sự kiện, $X = \left\{\{e_1\}, \{e_2\}, \{e_0\}, \{e_1, e_2\}, \{e_0, e_1\}, \{e_0, e_2\}, \{e_0, e_1, e_2\}\right\} \subset 2^A$ chứa tập hợp các trạng thái, $r(A, X)$ là mối quan hệ giữa sự kiện A và trạng thái X được biểu diễn bằng ma trận nhị phân ở hình 4.8.

4.3. Một số quan hệ nền tảng giữa trạng thái và sự kiện trong không gian Chu

Dựa vào *Tiểu mục 1.6.12* của *Chương 1*, luận án phân tích chi tiết thành bốn mối quan hệ nền tảng giữa hai sự kiện như sau:

Mối quan hệ độc lập ∇: Mối quan hệ độc lập $(e_0 \nabla e_1)$ mô tả việc thực thi độc lập của hai sự kiện e_0 và e_1 trong mọi trạng thái được phép. Nói cách khác, tất cả các tập con của A là các trạng thái có hiệu quả. Hình 4.9(a) trình bày ngữ nghĩa đại số và các công thức luận lý tương ứng của nó.

Mối quan hệ trước sau ≺: Mối quan hệ trước sau $(e_0 \prec e_1)$ biểu diễn sự kiện e_0 xảy ra trước sự kiện e_1. Nói cách khác, mối quan hệ trước sau này được sử dụng để đặc tả việc thực thi tuần tự của các sự kiện. Hình 4.9(b) trình bày ngữ nghĩa đại số và các công thức luận lý tương ứng của nó.

Mối quan hệ xung đột #: Mối quan hệ xung đột $(e_0 \# e_1)$ mô tả việc xảy ra của sự kiện e_0 hoặc sự kiện e_1. Nói cách khác, cả hai sự kiện e_0 và e_1 không thể xảy

ra cùng một lúc trong cùng một tính toán của ngữ nghĩa đại số. Hình 4.9(c) trình bày ngữ nghĩa đại số và công thức luận lý tương ứng của nó.

Mối quan hệ khả năng kích hoạt phân biệt den: Mối quan hệ khả năng kích hoạt phân biệt $\mathrm{den}(e_0, e_1, e_2)$ đặc tả sự xảy ra của sự kiện e_1 hoặc e_2 kích hoạt việc xảy ra của e_0. Hơn nữa, mối quan hệ này có thể được sử dụng cùng với mối quan hệ xung đột để đặc tả quyền cho các sự kiện theo các dòng của một câu lệnh $if\ldots\ then\ldots\ else$. Hình 4.9(d) trình bày cho ngữ nghĩa đại số và các công thức luận lý tương ứng của nó.

∇	e_1	e_1	$\boldsymbol{f}_{\nabla} =$
s_0	0	1	$\{\bar{e_0}, e_1\} +$
s_1	0	0	$\{\bar{e_0}, \bar{e_1}\} +$
s_2	1	1	$\{e_0, e_1\} +$
s_3	1	0	$\{e_0, \bar{e_1}\} = 1$

(a) Ma trận quan hệ độc lập

$\prec$	e_0	e_1	$\boldsymbol{f}_{\prec} =$
s_0	1	0	$\{e_0, \bar{e_1}\} +$
s_1	0	0	$\{\bar{e_0}, \bar{e_1}\} +$
s_2	1	1	$\{e_0, e_1\} = e_0 + \bar{e_1}$

(b) Ma trận quan hệ trước sau

den	e_0	e_1	e_2	$\boldsymbol{f}_{\mathrm{den}} =$
s_0	0	1	0	$\{\bar{e_0}, e_1, \bar{e_2}\} +$
s_1	0	0	0	$\{\bar{e_0}, \bar{e_1}, \bar{e_2}\} +$
s_2	0	1	1	$\{\bar{e_0}, e_1, e_2\} +$
s_3	0	0	1	$\{\bar{e_0}, \bar{e_1}, e_2\} +$
s_4	1	0	1	$\{e_0, \bar{e_1}, e_2\} +$
s_5	1	1	1	$\{e_0, e_1, e_2\} +$
s_6	1	1	0	$\{e_0, e_1, \bar{e_2}\} = \bar{e_0} + e_1 + e_2$

(d) Ma trận quan hệ khả năng kích hoạt phân tách

$\#$	e_0	e_1	$\boldsymbol{f}_{\#} =$
s_0	0	1	$\{\bar{e_0}, e_1\} +$
s_1	0	0	$\{\bar{e_0}, \bar{e_1}\} +$
s_2	1	0	$\{e_0, \bar{e_1}\} = \bar{e_0} + \bar{e_1}$

(c) Ma trận quan hệ xung đột

Hình 4.9. Một số quan hệ nền giữa các sự kiện và các trạng thái trong không gian Chu.

4.4. Ngữ nghĩa đại số dùng cho RTL

Luận án sử dụng một số phép toán xử lý stream dạng BDL trong các công trình [CT.1] và [CT.2] để tính toán stream trong thuật toán 4.2 *(Tiểu mục 4.4.3)* và thuật toán 4.4 *(Tiểu mục 4.6.1)*. Trong chương này, ngữ nghĩa đại số dùng cho đặc tả mối quan hệ giữa $\mathrm{ASA_{RTL}}$ và $\mathrm{ASA_{SPEC}}$ của thuật toán 4.2 ở hình 4.10(a) và thuật toán 4.4 ở hình 4.16(a) được phát triển để kiểm chứng các kết quả tổng hợp RTL.

4.4.1. Mô tả tập sự kiện

Định nghĩa 4.1 (Tính toán): Một dãy tuần tự các hành động hữu hạn được gọi là một quá trình tính toán trên tập các sự kiện A và tập hợp các tính toán được biểu diễn bằng ký hiệu $\mathcal{C}(A)$ [CT.4][CT.6].

4.4.2. Mô tả tập trạng thái

Giả sử X là tập hợp các trạng thái. Mỗi trạng thái $s \in X$ được định nghĩa bằng một mối quan hệ chuyển tiếp $T(e, s_i)$, nhiều khi mối quan hệ chuyển tiếp này cũng được biểu diễn theo dạng sau $s_i \xrightarrow{e} s_j$, trong đó sự kiện $e \in A$ có thể tạo thành một quan hệ chuyển tiếp từ trạng thái $s_i \in X$ sang trạng thái $s_j \in X$. Do đó, mỗi trạng thái s_j là một tập con của sự kiện A chứa các sự kiện có thể tạo ra một chuyển tiếp từ s_i, được hiểu phép chuyển tiếp có dạng như sau: $s_j = \left\{e \mid e \in A: s_i \xrightarrow{e} s_j\right\}$.

Nói một cách khác, bộ ba thành phần (A, X, T) này tạo thành một ngữ nghĩa đại số. Với mọi e_i trong A và s_i trong X, một tính chất tính toán của mô hình dựa trên ngữ nghĩa đại số như sau: $\mathrm{ASM}(A, X, T)$ có tính chất: $\mathcal{T} = \left\{e_1, e_2, e_3, \ldots, e_n \mid s_1 \xrightarrow{e_1} s_2, s_2 \xrightarrow{e_2} s_3, \ldots, s_n \xrightarrow{e_n} s_{n+1}\right\}$. Từ suy luận này, khái niệm về máy trạng thái đại số được phát triển trong các phần tiếp theo của chương 4 này [CT.4][CT.6].

4.4.3. Thuật toán 4.2 dùng cho stream dạng BDL

Thuật toán 4.2 được phân bổ danh sách sự kiện $e_0, e_1, \ldots, e_6$ cho các phép toán xử lý stream ($merge$, $drop$, $take$, zip, $split$ và $assign$) với một bước CStep đơn dụng trong thuật toán 4.1 (Chia thành CStep). Các bước tổng hợp đa dạng của tính toán BDL dựa trên phép tính stream được trình bày thông qua minh họa ánh xạ hai stream đầu vào $\sigma, \tau \in \mathbb{A}^\omega$ thành một stream đầu ra $\epsilon \in \mathbb{A}^\omega$.

Thuật toán 4.1: Xử lý tính toán 02 stream đầu vào bằng các phép toán xử lý stream *(phi vòng lặp)* cho ra 01 stream đầu ra và được gắn CStep.

Đầu vào: $\sigma, \tau \in \mathbb{A}^\omega$
Đầu ra : $\epsilon \in \mathbb{A}^\omega$

1.	**Begin**	**CStep**
2.	$\alpha := drop(\sigma + take(\tau))$;	
3.	$\gamma := split(\sigma) + zip(\sigma + take(\tau))$;	
4.	If $\alpha <> \gamma$ Then	
5.	$\epsilon := \alpha$;	CStep
6.	Else	
7.	$\epsilon := \gamma$;	
8.	End if;	
9.	Sending(ϵ);	
10.	**End**	

Thuật toán 4.2: Xử lý tính toán 02 stream đầu vào bằng các phép toán xử lý stream *(phi vòng lặp)* cho ra 01 stream đầu ra và được gắn sự kiện.

Đầu vào: $\sigma, \tau \in \mathbb{A}^\omega$
Đầu ra : $\epsilon \in \mathbb{A}^\omega$

1.	**Begin**		**Sự kiện**
2.	$\pi := \sigma + take(\tau)$;	_____	e_0
3.	$\alpha := drop(\pi)$;	_____	e_1
4.	$\gamma := split(\sigma) + zip(\pi)$;	_____	e_2
5.	If $\alpha <> \gamma$ Then	_____	e_3
6.	$\epsilon := \alpha$;	_____	e_4
7.	Else		
8.	$\epsilon := \gamma$;	_____	e_5
9.	End if;		
10.	Sending(ϵ);	_____	e_6
11.	**End**		

Xét tiến trình tổng hợp tính toán BDL dựa trên phép tính stream thông qua minh họa trên tính toán hai stream đầu vào $\sigma, \tau \in \mathbb{A}^{\omega}$ và cuối cùng ánh xạ các stream thành stream đầu ra $\epsilon \in \mathbb{A}^{\omega}$ được mô tả qua Thuật toán 4.2.

4.4.4. Máy trạng thái đại số của thuật toán 4.2

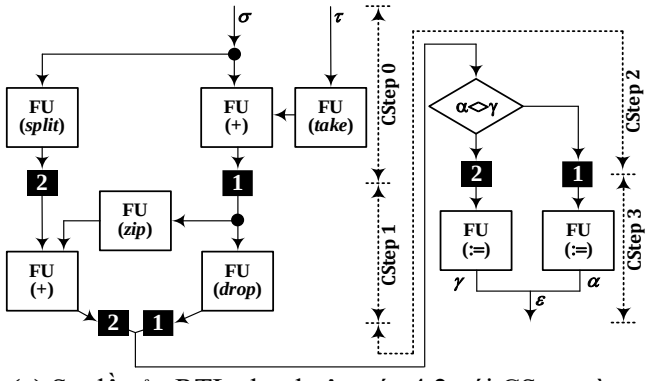
Định nghĩa 4.2 (Máy trạng thái đại số): Máy trạng thái đại số là bộ ba $\mathcal{A} = (X, s_0, T)$, với X là tập hữu hạn các trạng thái, s_0 là trạng thái khởi đầu, và T là một hàm chuyển tiếp từ $X \times A$ vào $X \cup \{\perp\}$ [CT.4][CT.6].

Nếu $T(s, e) = \perp$, không còn chuyển tiếp nào được gắn nhãn bởi sự kiện e từ trạng thái s. Trạng thái $\perp$ được gọi là trạng thái nhận vào. Một chuỗi tính toán $\mathcal{T} = \{e_1, e_2, e_3, \ldots, e_n\}$ được chấp nhận bởi máy trạng thái đại số nếu tồn tại $s_1, s_2, s_3, \ldots, s_n \in X$ sao cho $T(s_0, e_1) = s_1$, $T(s_{i-1}, e_i) = s_i$ với $1 < i \leq n$. Điều này được biểu diễn qua dãy tuần tự các trạng thái được chuyển tiếp $s_0 \xrightarrow{e_1} s_1 \xrightarrow{e_2} s_2 \ldots s_{n-1} \xrightarrow{e_n} s_n$. Ngược lại, nếu tồn tại $1 \leq k \leq n$ sao cho $s_0 \xrightarrow{e_1} s_1 \xrightarrow{e_2} s_2 \ldots s_{k-1} \xrightarrow{e_k} \perp$, đường đi này là một stream dạng BDL của máy trạng thái qua tính toán $\mathcal{T}$. Tập hợp các tính toán được chấp nhận bởi $\mathcal{A}$ ký hiệu là $\mathcal{C}(\mathcal{A})$ (Định nghĩa 4.1). Bốn máy trạng thái đại số của bốn CStep, mỗi máy có bốn trạng thái, được minh họa ở các hình 4.10(a), 4.10(b), và 4.10(c).

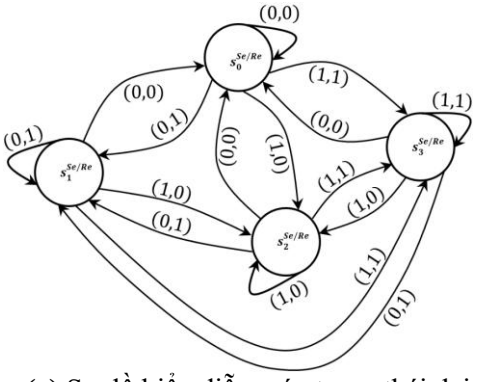
Các FUs thường được tích hợp vào một thư viện đại diện cho sự tương ứng giữa chức năng của chúng và các phép toán liên quan. Các FUs có thể thực hiện các phép toán đơn nhiệm hoặc các phép toán đa nhiệm với các tín hiệu đầu vào được kiểm soát để lựa chọn phép toán theo mong muốn. Trong hình 4.10(a), tám FUs đơn nhiệm là cần thiết để thực hiện chín phép toán sau: splitting, taking, zipping, dropping, merging, copying, assignment, so sánh và registering [CT.1][CT.2].

Trong hình 4.10(a) biểu diễn RTL của thuật toán 4.2 với các CStep. CStep 0 thực hiện bốn phép toán như sau: taking, merging, splitting và copying. Sau đó, kết quả được lưu trữ trong hai thanh ghi 1 và 2. Tiếp theo, CStep 1 thực hiện ba phép toán như sau: zipping, merging và dropping sử dụng giá trị của hai thanh ghi 1 và 2. Sau đó, kết quả được lưu trữ trong hai thanh ghi được sử dụng lại 1 và 2. CStep 2 thực hiện một so sánh sử dụng giá trị của hai thanh ghi 1 và 2 của CStep 1. Sau đó, kết quả được lưu trữ trong các thanh ghi được sử dụng lại 1 hoặc 2. Cuối cùng, CStep 3 thực hiện một phép toán gán sử dụng giá trị của hai thanh ghi 1 hoặc 2 từ CStep 2, sau đó gửi kết quả của phép toán gán cho đầu ra. Trong hình 4.10(b) biểu đạt cho ma trận với bốn CStep và thanh ghi tương ứng, đặc biệt là biểu diễn cho Trạng thái 0 $\left(s_0^{Se/Re}\right)$ đến Trạng thái 3 $\left(s_3^{Se/Re}\right)$ của CStep 0 với hai thanh ghi và CStep 1, 2 và 3 cũng được biểu diễn như vậy. Trong hình 4.10(c), hình này là một sơ đồ trực quan

mô tả ma trận trong hình 4.10(b) biểu diễn máy trạng thái đại số cho thuật toán 4.2 với bốn CStep.



(a) Sơ đồ của RTL cho thuật toán 4.2 với CStep cùng với FUs và các phép toán



(c) Sơ đồ biểu diễn máy trạng thái đại số cho thuật toán 4.2 với bốn CStep

	Thanh ghi			Thanh ghi			Thanh ghi			Thanh ghi	
CStep 0	1	2	**CStep 1**	1	2	**CStep 2**	1	2	**CStep 3**	1	2
$0\left(s_0^{Se/Re}\right)$	0	0	$0\left(s_0^{Re/Se}\right)$	0	0	$0\left(s_0^{Re/Se}\right)$	0	0	$0\left(s_0^{Re/Se}\right)$	0	0
$1\left(s_1^{Se/Re}\right)$	0	1	$1\left(s_1^{Re/Se}\right)$	0	1	$1\left(s_1^{Re/Se}\right)$	0	1	$1\left(s_1^{Re/Se}\right)$	0	1
$2\left(s_2^{Se/Re}\right)$	1	0	$2\left(s_2^{Re/Se}\right)$	1	0	$2\left(s_2^{Re/Se}\right)$	1	0	$2\left(s_2^{Re/Se}\right)$	1	0
$3\left(s_3^{Se/Re}\right)$	1	1	$3\left(s_3^{Re/Se}\right)$	1	1	$3\left(s_3^{Re/Se}\right)$	1	1	$3\left(s_3^{Re/Se}\right)$	1	1

(b) Các ma trận biểu diễn máy trạng thái đại số của thuật toán 4.2 với bốn CStep

Hình 4.10. Biểu diễn của máy trạng thái đại số cho thuật toán 4.2 với CStep

Tiếp theo, chương này sẽ giới thiệu tích máy trạng thái đại số trong phần 4.4.5 cho phép một quy trình mô đun hóa phức tạp hơn trong quá trình đặc tả. Trong đó, mỗi tiến trình con có thể được mô hình hóa bằng máy trạng thái đại số và việc mô hình hóa quá trình toàn diện có thể đạt được bằng cách tính toán tích máy trạng thái đại số của tất cả các chi tiết tiến trình con được mô tả trong phần tiếp theo.

4.4.5. Tích máy trạng thái đại số của thuật toán 4.2

Luận án trình bày hai tiến trình Se và Re có thể được kết nối trong hình 4.11(a) và hình 4.11(b). Ký hiệu $\mathcal{S}e = \left(X^{Se}, s_p^{Se}, T^{Se}\right)$ và $\mathcal{R}e = \left(X^{Re}, s_q^{Re}, T^{Re}\right)$ tương ứng với hai máy trạng thái đại số đã mô hình hóa hai tiến trình Se và Re này. Định nghĩa tích máy trạng thái đại số của hai tiến trình Se và Re là $\mathcal{S}e \times \mathcal{R}e$, để mô hình hóa tiến trình đạt được bằng cách kết nối Se với Re. Đối với việc không đồng bộ hóa đầu ra của Se với đầu vào của Re, khi dữ liệu di chuyển giữa Se và Re qua một thanh ghi thì sự chuyển giao này phải xảy ra. Điều này thể hiện xác định của tích $\mathcal{S}e$ và $\mathcal{R}e$ trên tập sự kiện A tương tự như $A{:}\,\mathcal{S}e \times \mathcal{R}e = \left(X, s_p, T\right)$, với $X = X^{Se} \times X^{Re}$, $s_p = \left(s_p^{Se} \times s_q^{Re}\right)$, trong đó p, q được thay thế bằng i, j, k, hoặc l.

Quan hệ chuyển đổi T được định nghĩa: Với $s_i = \left(s_i^{Se} \times s_j^{Re}\right) \in X$ và $e \in A$, nếu tồn tại $s_{i+1}^{Se} \in X^{Se}$ và $s_{j+1}^{Re} \in X^{Re}$ sao cho $T^{Se}(s_i^{Se}, e) = s_{i+1}^{Se}$ và $T^{Re}\left(s_j^{Re}, e\right) =$

s_{j+1}^{Re} thì $T\left(\left(s_i^{Se}, s_j^{Re}\right), e\right) = (s_{i+1}^{Se}, s_{j+1}^{Re})$. Ngược lại, $T\left(\left(s_i^{Se}, s_j^{Re}\right), e\right) = \perp$.

Giả sử rằng đầu ra/đầu vào của Se là tương ứng với đầu vào/đầu ra của Re, vậy thì các tiến trình này kết nối lẫn nhau như trong hình 4.11, trong đó hình 4.11(a) là một ma trận biểu diễn cho mười sáu trạng thái của bốn thanh ghi và sau đó hình 4.11(b) là một sơ đồ trực quan biểu diễn các RTL của 16 trạng thái với bốn thanh ghi. Mỗi trạng thái trong $\mathcal{S}e \times \mathcal{R}e$ là một cặp bao gồm một trạng thái từ $\mathcal{S}e$ và một trạng thái từ $\mathcal{R}e$. Việc chạy của tích máy trạng thái đại số $\mathcal{S}e \times \mathcal{R}e$ qua chuỗi tính toán được chấp nhận $\mathcal{T} = \{(1,1,0,1)(1,1,1,0)(1,1,1,1)\}$ được biểu diễn như sau:

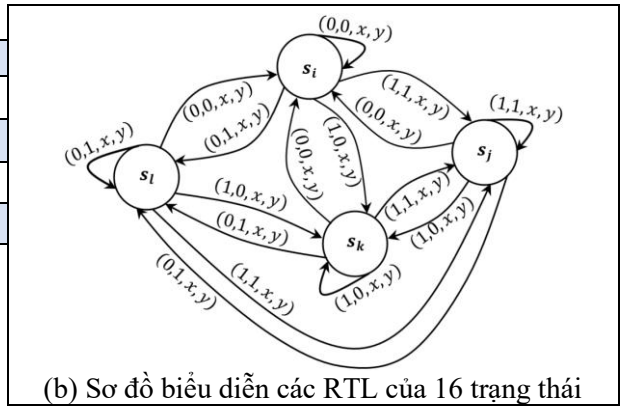
$$(s_3^{Se}, s_0^{Re}) \xrightarrow{(1,1,0,1)} (s_3^{Se}, s_1^{Re}) \xrightarrow{(1,1,1,0)} (s_3^{Se}, s_2^{Re}) \xrightarrow{(1,1,1,1)} (s_3^{Se}, s_3^{Re})$$

Điều này chỉ ra rằng trong trạng thái (s_3^{Se}, s_0^{Re}) trên sự kiện $(1,1,0,1)$, tích máy trạng thái đại số $\mathcal{S}e \times \mathcal{R}e$ được thực hiện bằng cách thực thi $\mathcal{S}e$ từ s_3^{Se} và đồng thời thực thi $\mathcal{R}e$ từ s_0^{Re} và đồng thời thực thi $\mathcal{S}e$ và cứ làm như vậy.

	Các thanh ghi			
$\mathcal{S}e \times \mathcal{R}e$	1	2	3	4
$s_i = \left(s_i^{Se} \times s_j^{Re}\right)$	0	0	x	y
$s_j = \left(s_j^{Se} \times s_k^{Re}\right)$	0	1	x	y
$s_k = \left(s_k^{Se} \times s_l^{Re}\right)$	1	0	x	y
$s_l = \left(s_l^{Se} \times s_i^{Re}\right)$	1	1	x	y

Trong đó: $x, y \in \{0,1\}$ và $x \neq y$;
$i, j, k, l = [\overline{0,3}] \subset \mathbb{N}^* = \mathbb{N} \cup (0)$

(a) Ma trận này biểu diễn 16 trạng thái của 4 thanh ghi



(b) Sơ đồ biểu diễn các RTL của 16 trạng thái

Hình 4.11. Sơ đồ kết nối của tiến trình Se và Re và tích máy trạng thái đại số của nó

4.4.6. Sự tương đương của các thuật toán

Cho thuật toán 4.2 và 4.2′ là hai thuật toán và $\mathcal{A}$ và $\mathcal{A}'$ là hai máy trạng thái đại số tương ứng của chúng. Thuật toán 4.2 và 4.2′ là tương đương $\Leftrightarrow \mathcal{C}(\mathcal{A}) = \mathcal{C}(\mathcal{A}')$.

4.4.7. Ngữ nghĩa đại số của thuật toán 4.2

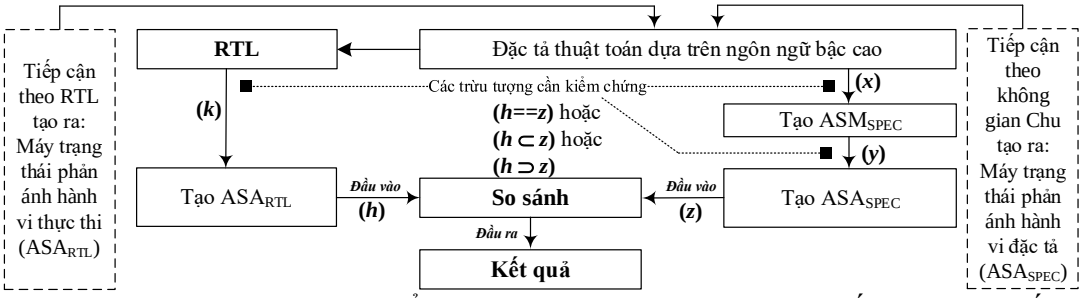
Một ngữ nghĩa đại số (một khía cạnh đại số) có thể được giải thích như một máy trạng thái đại số với tập sự kiện A và tập trạng thái có thể có X. Khi chuyển đến một trạng thái s, các máy trạng thái đại số thực thi một số chuyển tiếp qua các sự kiện để đạt đến một trạng thái thay thế của s. Mỗi tính chất tính toán có thể có của ngữ nghĩa đại số tương ứng với mỗi việc chạy của máy trạng thái đại số và việc thực thi của ngữ nghĩa đại số chỉ định ra tập chạy của máy trạng thái đại số.

Máy trạng thái đại số trong ngữ cảnh của một mô hình ngữ nghĩa đại số được biểu diễn bởi tập sự kiện A và các sự kiện xảy ra cho mỗi trạng thái, có thể được hiểu là mối quan hệ chuyển tiếp T. Một cách tiếp cận thực tế hơn dựa vào các mối quan hệ giữa các trạng thái là mỗi máy trạng thái đại số được mô hình hóa như một tập mối quan hệ giữa các trạng thái và đối với mỗi mối quan hệ như vậy phần này sẽ có một

ngữ nghĩa đại số tương ứng. Ngữ nghĩa đại số của mỗi mối quan hệ giữa các trạng thái được xem như là một tính chất của máy trạng thái đại số. Do vậy, sự kết hợp của các tính chất sẽ dẫn đến máy trạng thái đại số.

4.5. Thuật toán 4.2 cho các kết quả tổng hợp RTL

4.5.1. Các bước thuật toán kiểm chứng



Hình 4.12. Các bước thuật toán kiểm chứng dựa trên ngữ nghĩa đại số cho ra các kết quả tổng hợp RTL, phục vụ kiểm chứng logic hệ thống livestream

Các bước của thuật toán kiểm chứng được biểu diễn qua sơ đồ trong hình 4.12. Các bước của thuật toán được thể hiện cụ thể như hình 4.12 ở trên.

4.5.2. Đặc tả thuật toán dựa trên ngôn ngữ mức cao

Mô tả chi tiết về đặc tả thuật toán dựa trên ngôn ngữ mức cao được xem là thích hợp để mô tả, điều này có nghĩa là một chương trình được tạo ra trong bước này. Nhiệm vụ chính trong quá trình này là cập nhật các kiểu của việc lập lịch khác nhau. Do đó, trong minh họa về ngôn ngữ mô tả mức cao được trình bày trong thuật toán 4.2 với bảy sự kiện $e_0, e_1, e_2, ..., e_6$ liên quan đến các câu lệnh, ở đó mỗi câu lệnh được xem như là một sự kiện.

4.5.3. Tạo mô hình dựa trên ngữ nghĩa đại số (ASM)

Vấn đề chính là tạo ra những đặc tả thứ tự cục bộ cho các câu lệnh chương trình được tạo ra ở tiểu mục 4.4.3. Nói cách khác, đặc tả này được sử dụng để thể hiện sự phụ thuộc dữ liệu giữa các câu lệnh cần thiết cho việc tạo ra mô hình dựa trên ngữ nghĩa đại số được gọi là $\mathrm{ASM_{SPEC}}$. Để thực hiện công việc này, ta cần tham chiếu đến một số mối quan hệ nền tảng giữa các sự kiện trong tiểu mục 4.3 ở trên.

Tiếp cận của chương này để tạo ra $\mathrm{ASM_{SPEC}}$, luận án sử dụng các mối quan hệ giữa các sự kiện ở tiểu mục 4.3. Các mối quan hệ này có thể được trích xuất từ đặc tả hệ thống được cung cấp trong một ngôn ngữ lập trình mức cao như trong mục 4.5.2. Giả sử có một thuật toán 4.2 với tập sự kiện A, cùng với các mối quan hệ r giữa các sự kiện, được trích xuất từ mô tả thuật toán 4.1. Thuật toán 4.2 có bảy sự kiện, trong đó e_0 xảy ra trước e_1, e_0 xảy ra trước e_2, e_1 độc lập với e_2, e_1 xảy ra trước e_3, e_2 xảy ra trước e_3, e_3 xảy ra trước e_4, e_3 xảy ra trước e_5, e_4 và e_5 xảy ra xung đột, và việc thực thi của e_4 hoặc e_5 kích hoạt sự xuất hiện của e_6. Do đó, sự kết hợp này mang lại một mô hình ngữ nghĩa đại số (ASM) của Thuật toán 4.2 dùng cho BDL. Một cách

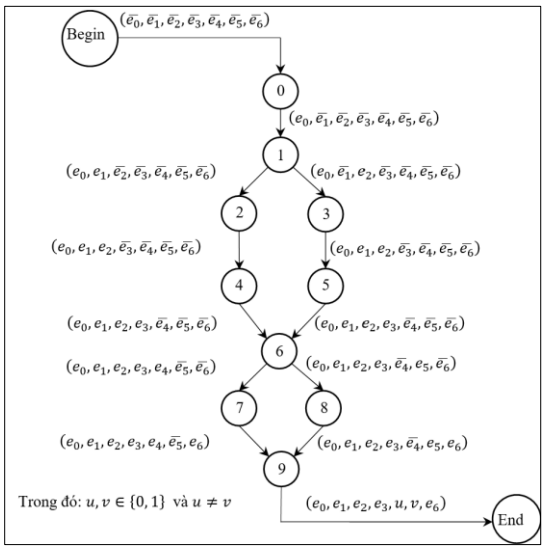
hình thức hơn thì ASM_{SPEC} được mô tả như sau: $\mathbf{ASM_{SPEC}} = \{e_0 \prec e_1, e_0 \prec e_2, e_1 \nabla e_2, e_1 \prec e_3, e_2 \prec e_3, e_3 \prec e_4, e_3 \prec e_5, e_4 \# e_5, \mathbf{den}(e_6, e_4, e_5)\}$

Từ danh sách các sự kiện của thuật toán 4.2 và dựa vào mối quan hệ phụ thuộc giữa các sự kiện được tạo ra trong ASM_{SPEC}, một máy trạng thái đặc tả đại số của ASM_{SPEC} được tạo ở hình 4.13.

ASA_{SPEC}	e_0	e_1	e_2	e_3	e_4	e_5	e_6	Công thức logic (ASM_{SPEC})
s_0	0	0	0	0	0	0	0	$f = f_{e_0 \prec e_1} \wedge f_{e_0 \prec e_2} \wedge f_{e_1 \nabla e_2} \wedge f_{e_1 \prec e_3} \wedge f_{e_2 \prec e_3} \wedge f_{e_3 \prec e_4} \wedge f_{e_3 \prec e_5} \wedge f_{e_4 \# e_5} \wedge f_{\mathrm{den}(e_6,e_4,e_5)} =$ $(e_0 + \bar{e_1}) \wedge (e_0 + \bar{e_2}) \wedge 1 \wedge (e_1 + \bar{e_3}) \wedge (e_2 + \bar{e_3}) \wedge (e_3 + \bar{e_4}) \wedge (e_3 + \bar{e_5}) \wedge (\bar{e_4} + \bar{e_5}) \wedge (\bar{e_6} + e_4 + e_5) =$ $\bar{e_1}\bar{e_2}\bar{e_3}\bar{e_4}\bar{e_5}\bar{e_6} + e_0\bar{e_3}\bar{e_4}\bar{e_5}\bar{e_6} + e_0e_2\bar{e_3}\bar{e_4}\bar{e_5}\bar{e_6}$ $+e_0e_1\bar{e_3}\bar{e_4}\bar{e_5}\bar{e_6} + e_0e_1e_2\bar{e_4}\bar{e_5}\bar{e_6}\ e_0e_1e_2e_3\bar{e_4}\bar{e_6}$ $+e_0e_1e_2e_3\bar{e_4}e_5 + e_0e_1e_2e_3\bar{e_5}\bar{e_6}\ e_0e_1e_2e_3e_4\bar{e_5}$
s_1	1	0	0	0	0	0	0	
s_2	1	0	1	0	0	0	0	
s_3	1	1	0	0	0	0	0	
s_4	1	1	1	0	0	0	0	
s_5	1	1	1	1	0	0	0	
s_6	1	1	1	1	0	1	0	
s_7	1	1	1	1	0	1	1	
s_8	1	1	1	1	1	0	0	
s_9	1	1	1	1	1	0	1	

Hình 4.13. ASA_{SPEC} của thuật toán 4.2

4.5.4. Kết quả tổng hợp RTL



Hình 4.14. ASA của thuật toán 4.2

ASA_{RTL}	e_0	e_1	e_2	e_3	e_4	e_5	e_6
s_0	0	0	0	0	0	0	0
s_1	1	0	0	0	0	0	0
s_2	1	0	1	0	0	0	0
s_3	1	1	0	0	0	0	0
s_4	1	1	1	0	0	0	0
s_5	1	1	1	1	0	0	0
s_6	1	1	1	1	1	0	0
s_7	1	1	1	1	0	1	0
s_8	1	1	1	1	0	1	1
s_9	1	1	1	1	1	0	1
s_{10}	1	1	1	1	u	v	1
Trong đó: $u, v \in \{0,1\}$ và $u \neq v$							

Hình 4.15. ASA_{RTL} của thuật toán 4.2

Sau đây là kết quả tổng hợp RTL của đặc tả hành vi thuật toán, kết quả này được tạo ra từ giai đoạn tổng hợp và được chuyển sang giai đoạn kiểm chứng. Mô đun RTL [CT.2] được định nghĩa. Đơn vị điều khiển được tạo thành từ các bước. Mỗi bước được đánh số và phải được thực thi trong một đơn vị đồng hồ duy nhất. Hình 4.14 đã thể hiện đơn vị điều khiển này cho thuật toán 4.2:

- Đơn vị xử lý chứa khai báo về các thành phần cấu thành đơn vị xử lý.
- Đơn vị điều khiển xác định dãy tuần tự lệnh cục bộ cần phát ra bởi đơn vị điều khiển.
- Gán cố định xác định một hoạt động cần được lặp lại trong mỗi chu kỳ đồng hồ.

4.5.5. Tạo máy trạng thái RTL đại số

ASA_{RTL} (Hình 4.15) được tạo ra từ kết quả của tiến trình tổng hợp RTL. Do đó, từ tiến trình tổng hợp RTL của thuật toán 4.2 như đã được thể hiện ở hình 4.14 dẫn

đến ASA_{RTL} được tạo ra ở hình 4.15.

4.6. Ngữ nghĩa đại số dùng cho RTL

4.6.1. Thuật toán 4.4 dùng cho stream dạng BDL

Thuật toán 4.3: Xử lý tính toán 01 stream đầu vào bằng các phép toán xử lý stream *(với vòng lặp)* cho ra 01 stream đầu ra và được gắn CStep.

Đầu vào : $\sigma \in \mathbb{A}^{\omega}$
Đầu ra : $\gamma \in \mathbb{A}^{\omega}$

CStep

1. Begin
2. $a[0] := take(\sigma)$;
3. $\gamma := drop(a[0])$;
4. $a[1] := split(a[0])$;
5. $\gamma := rev(a[1])$;
6. $a[2] := zip(a[1] + a[0])$;
7. $\gamma := drop(a[2])$;
8. $i := 3$;
9. If ($i \leq 5$)
10. $a[i] := split(a[i-1]) + drop(a[i-2] + a[i-3])$;
11. $\gamma := a[i]$;
12. $i := i + 1$;
13. goto(10);
14. End if;
15. Sending(γ);
16. End

CStep

Thuật toán 4.4: Xử lý tính toán 01 stream đầu vào bằng các phép toán xử lý stream *(với vòng lặp)* cho ra 01 stream đầu ra và được gắn các sự kiện.

Đầu vào : $\sigma \in \mathbb{A}^{\omega}$
Đầu ra : $\gamma \in \mathbb{A}^{\omega}$

			Sự kiện
1.	**Begin**		
2.	$a[0] := take(\sigma)$;	_	e_0
3.	$\gamma := drop(a[0])$;	_	e_1
4.	$a[1] := split(a[0])$;	_	e_2
5.	$\gamma := rev(a[1])$;	_	e_3
6.	$a[2] := zip(a[1] + a[0])$;	_	e_4
7.	$\gamma := drop(a[2])$;	_	e_5
8.	$i := 3$;	_	e_6
9.	If ($i \leq 5$)	_	e_7
10.	$a[i] := split(a[i-1]) + drop(a[i-2] + a[i-3])$;	_	e_8
11.	$\gamma := a[i]$;	_	e_9
12.	$i := i + 1$;	_	e_{10}
13.	goto(10);	_	e_{11}
14.	End if;		
15.	Sending(γ);	_	e_{12}
16.	**End**		

Thuật toán 4.4 phân bổ danh sách các sự kiện cho các phép toán stream ($merge$, $drop$, $take$, zip, $split$, và $assign$) với một CStep duy nhất. Các bước tổng hợp tính toán BDL dựa trên phép tính stream được mô tả thông qua minh họa ánh xạ các stream

đầu vào $\sigma \in \mathbb{A}^{\omega}$ sang các bộ tổ hợp stream đầu ra thành $\gamma \in \mathbb{A}^{\omega}$ như được định nghĩa thuật toán 4.4.

Xét tổng hợp tính toán BDL dựa trên phép tính stream thông qua minh họa tính toán một mảng các stream dạng BDL, cụ thể là mảng a bao gồm một phép toán assign kết hợp một số phép toán *take, split, zip, drop* và *merge* trên BDL như sau: $a[0] := take(\sigma)$, $a[1] := split(a[0])$, $a[2] := zip(a[1] + a[0])$, với số nguyên bắt đầu $i = 3$ và số lần lặp tùy ý là i nhưng kết thúc lặp khi $i \leq 5$, cung cấp $a[i] := split(a[i-1]) + drop(a[i-2] + a[i-3])$ và ánh xạ các stream vào đầu ra γ được xác định bởi thuật toán 4.4. Xét tổng hợp tính toán BDL dựa trên phép tính stream thông qua minh họa này để tính toán stream đầu vào $\sigma \in \mathbb{A}^{\omega}$, và cuối cùng ánh xạ các stream sang đầu ra γ được xác định bởi thuật toán 4.4.

4.6.2. Máy trạng thái đại số

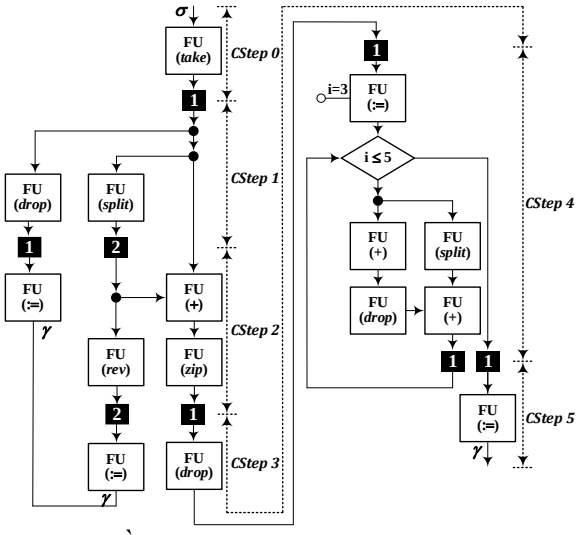
Hình 4.16, một đường đi thông qua máy trạng thái như vậy được gọi là chạy tự động hóa qua phép tính $\mathcal{T}$. Tập các phép tính chấp nhận bởi một bộ ba $\mathcal{A}$ được biểu diễn bằng ký hiệu $\mathcal{C}(\mathcal{A})$ qua định nghĩa 4.1. Bốn máy trạng thái đại số của sáu CSteps bao gồm bốn trạng thái trong CStep 1 và CStep 2, hai trạng thái trong các CStep 0, 3, 4 và 5 được thể hiện chi tiết ở hình 4.16(a), 4.16(b) và Hình 4.16(c). Trong ngữ cảnh thiết kế các hệ thống, các đơn vị chức năng là các thành phần chính thực hiện các phép toán hoặc các nhiệm vụ cụ thể. Thông thường, chúng được tích hợp vào một thư viện biểu diễn ánh xạ giữa chức năng của chúng và các phép toán liên quan [CT.1][CT.2]. Các đơn vị chức năng có thể là các triển khai hóa của:

- *Các phép toán đơn dụng:* Được thiết kế để thực hiện một phép toán cụ thể. Minh họa một bộ trộn là một phép toán đơn dụng thực hiện việc trộn.
- *Các phép toán đa dụng:* Đa dạng hơn và có thể thực hiện nhiều phép toán khác nhau. Thông thường, chúng có các tín hiệu điều khiển đầu vào cho phép chọn lựa phép toán mong muốn. Minh họa một đơn vị logic toán học (ALU) có thể thực hiện các chức năng khác nhau như merging, splitting, và phép toán logic tùy thuộc vào các tín hiệu điều khiển mà nó nhận được.

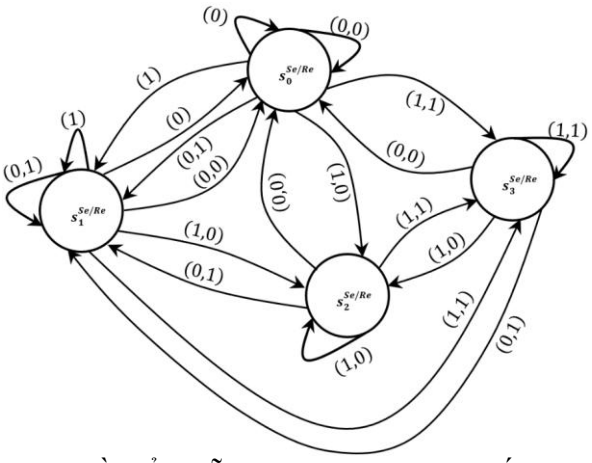
Tiếp cận mô đun hóa này cho phép linh hoạt và hiệu quả hơn trong thiết kế hệ thống. Nó cho phép các nhà thiết kế xây dựng các hệ thống phức tạp bằng cách kết hợp các đơn vị chức năng này theo nhiều cách khác nhau. Trong hình 4.16(a), có mười lăm đơn vị chức năng đơn dụng được yêu cầu để triển khai mười phép toán sau: $split$, $take$, zip, rev, $drop$, merge $+$, $copy$ $\bullet$, $assign$ $:=$, $\leq$, và $register$ $\blacksquare$.

Hình 4.16(a) biểu diễn RTL của thuật toán 4.4 với các CStep. CStep 0 thực hiện một phép toán taking. Sau đó, kết quả được lưu trữ trong thanh ghi số 1. Tiếp theo, CStep 1 thực hiện ba phép toán như sau: $copy$, $drop$ và $split$ bằng việc sử dụng các

giá trị của thanh ghi số 1. Sau đó, kết quả được lưu trữ trong hai thanh ghi số 1 và 2. Tiếp theo, CStep 2 thực hiện năm phép toán như sau: $assign$, $copy$, $merge$, $reverse$, và zip bằng việc sử dụng các giá trị của hai thanh ghi số 1 và 2. Sau đó, kết quả được lưu trữ trong việc tái sử dụng lại hai thanh ghi số 1 và 2. Tiếp theo, CStep 3 thực hiện hai phép toán như sau: $assign$ và $drop$ bằng việc sử dụng các giá trị của hai thanh ghi số 1 và 2. Sau đó, kết quả được lưu trữ trong việc tái sử dụng lại trong thanh ghi số 1. Tiếp theo, CStep 4 thực hiện năm phép toán như sau: $assign$, so sánh nhỏ hơn hoặc bằng, $merge$, $take$ và $drop$ bằng việc sử dụng các giá trị của thanh ghi số 1. Sau đó, kết quả được lưu trữ trong việc tái sử dụng lại trong thanh ghi số 1. Cuối cùng, CStep 5 thực hiện một phép toán $assign$ trong việc tái sử dụng giá trị của thanh ghi số 1 từ CStep 4, sau đó gửi kết quả của phép toán $assign$ ra ngoài.



(a) Sơ đồ RTL của thuật toán 4.4 với sáu CStep cùng với các FUs và các phép toán



(c) Sơ đồ biểu diễn máy trạng thái đại số cho các RTL với sáu CStep

	Thanh ghi
CStep 0	**1**
Trạng thái 0 $(s_0^{Se/Re})$	0
Trạng thái 1 $(s_1^{Se/Re})$	1

	Thanh ghi	
CStep 1	**1**	**2**
Trạng thái 0 $(s_0^{Re/Se})$	0	0
Trạng thái 1 $(s_1^{Re/Se})$	0	1
Trạng thái 2 $(s_2^{Re/Se})$	1	0
Trạng thái 3 $(s_3^{Re/Se})$	1	1

CStep 2	**2**	**1**
Trạng thái 0 $(s_0^{Re/Se})$	0	0
Trạng thái 1 $(s_1^{Re/Se})$	0	1
Trạng thái 2 $(s_2^{Re/Se})$	1	0
Trạng thái 3 $(s_3^{Re/Se})$	1	1

	Thanh ghi
CStep 3	**1**
Trạng thái 0 $(s_0^{Re/Se})$	0
Trạng thái 1 $(s_1^{Re/Se})$	1

	Thanh ghi
CStep 4	**1**
Trạng thái 0 $(s_0^{Re/Se})$	0
Trạng thái 1 $(s_1^{Re/Se})$	1

	Thanh ghi
CStep 5	**1**
Trạng thái 0 $(s_0^{Re/Se})$	0
Trạng thái 1 $(s_1^{Re/Se})$	1

(b) Các ma trận biểu diễn máy trạng thái đại số của thuật toán 4.4 với sáu CStep

Hình 4.16. Biểu diễn máy trạng thái đại số cho thuật toán 4.4 với sáu CStep

Hình 4.16(b) biểu diễn ma trận với sáu CStep và các thanh ghi tương ứng, cụ

thể là biểu diễn cho trạng thái 0 $\left(s_0^{Se/Re}\right)$ đến trạng thái 1 $\left(s_1^{Se/Re}\right)$ của CStep 0 với một thanh ghi, trạng thái 0 $\left(s_0^{Se/Re}\right)$ đến trạng thái 3 $\left(s_3^{Se/Re}\right)$ của CStep 1 với hai thanh ghi số 1 và 2 và tương tự CStep 2 cũng được biểu diễn như vậy, trạng thái 0 $\left(s_0^{Se/Re}\right)$ đến trạng thái 1 $\left(s_1^{Se/Re}\right)$ của CStep 3 với một thanh ghi và tương tự CStep 4 và 5 cũng được biểu diễn như vậy. Hình 4.16(c) là một sơ đồ mô tả ma trận Hình 4.16(b) biểu diễn máy trạng thái đại số cho thuật toán 4.4 với sáu CStep.

Chương này trình bày một tích máy trạng thái đại số trong phần 4.6.3 cho phép đặc tả mô đun hóa tiến trình phức tạp hơn sẽ được đặc tả cụ thể. Trong đó, mỗi tiến trình con có thể được mô hình hóa bằng máy trạng thái đại số và tính đầy đủ của tiến trình mô hình hóa có thể đạt được bằng cách tính toán tích máy trạng thái đại số của hầu hết các tiến trình con.

4.6.3. Tích máy trạng thái đại số của thuật toán 4.4

Biểu diễn hai tiến trình gửi Se và nhận Re có thể được liên kết với nhau như trong hình 4.17(a) và 4.17(b). Cho hai tiến trình $\mathcal{S}e = \left(X^{Se}, s_p^{Se}, T^{Se}\right)$ và $\mathcal{R}e = \left(X^{Re}, s_q^{Re}, T^{Re}\right)$ tương ứng hai máy trạng thái để mô hình hóa tiến trình Se và Re.

Giả sử rằng đầu vào hoặc đầu ra của Se là tương đương với đầu ra hoặc đầu vào của Re, vậy các tiến trình này có thể được liên kết với nhau như trong hình 4.17, trong đó hình 4.17(a) là một ma trận biểu diễn cho ba mươi hai trạng thái của năm thanh ghi và sau đó hình 4.17(b) là một sơ đồ trực quan biểu diễn cho các RTL của ba mươi hai trạng thái với năm thanh ghi. Mỗi trạng thái trong $\mathcal{S}e \times \mathcal{R}e$ là một cặp bao gồm một trạng thái từ $\mathcal{S}e$ và một trạng thái từ $\mathcal{R}e$. Việc chạy của tích máy trạng thái đại số của $\mathcal{S}e$ và $\mathcal{R}e$ dưới dạng sau $\mathcal{S}e \times \mathcal{R}e$ qua tính toán được chấp nhận $\mathcal{T} = \{(1,1,1,0,0)(1,1,1,0,1)(1,1,1,1,0)\}$ thể hiện như sau:

$$(s_3^{Se}, s_0^{Re}) \xrightarrow{(1,1,1,0,0)} (s_3^{Se}, s_1^{Re}) \xrightarrow{(1,1,1,0,1)} (s_3^{Se}, s_2^{Re}) \xrightarrow{(1,1,1,1,0)} (s_3^{Se}, s_3^{Re})$$

Trong trạng thái (s_3^{Se}, s_0^{Re}) với sự kiện $(1,1,1,0,0)$, tích máy trạng thái đại số $\mathcal{S}e \times \mathcal{R}e$ được thực hiện bằng cách thực thi $\mathcal{S}e$ từ trạng thái s_3^{Se} và song song thực thi $\mathcal{R}e$ từ trạng thái s_0^{Re} và song song thực thi $\mathcal{S}e$ và cứ tiếp tục như vậy.

4.6.4. Sự tương đương của các thuật toán

Cho thuật toán 4.4 và 4.4′ là hai thuật toán vòng lặp và $\mathcal{A}$ và $\mathcal{A}'$ là hai máy trạng thái đại số tương ứng của chúng. Thuật toán 4.4 và 4.4′ tương đương $\Leftrightarrow$ $\mathcal{C}(\mathcal{A}) = \mathcal{C}(\mathcal{A}')$.

4.6.5. Ngữ nghĩa đại số của thuật toán 4.4

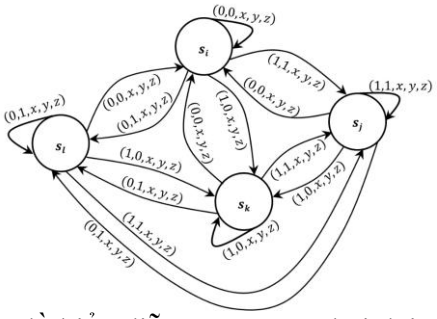
Ngữ nghĩa đại số được xem như khía cạnh đại số có thể được diễn giải như máy trạng thái đại số với tập các sự kiện A và tập trạng thái có thể có X. Khi chuyển đến một trạng thái s, các máy trạng thái đại số thực thi một số chuyển tiếp qua các sự kiện

để đạt đến một trạng thái thay thế của s. Mỗi tính toán có thể có của ngữ nghĩa đại số tương ứng với từng việc chạy của các máy trạng thái đại số và việc thực thi ngữ nghĩa đại số chỉ định ra tập chạy của các máy trạng thái đại số.

Máy trạng thái đại số trong một mô hình ngữ nghĩa đại số được biểu diễn bằng tập sự kiện A và các sự kiện xảy ra với mỗi trạng thái là quan hệ chuyển tiếp T. Một cách tiếp cận thực tế hơn dựa trên các mối quan hệ giữa các trạng thái, mỗi máy trạng thái đại số được mô hình hóa như một tập hợp các quan hệ giữa các trạng thái và đối với mỗi quan hệ như vậy có một ngữ nghĩa đại số tương ứng. Ngữ nghĩa đại số của mỗi quan hệ giữa các trạng thái được xem như một tính chất của máy trạng thái đại số. Do đó, sự kết hợp của các tính chất dẫn đến máy trạng thái đại số.

	Các thanh ghi				
$\mathcal{S}e \times \mathcal{R}e$	1	2	3	4	5
$s_i = \left(s_i^{Se} \times s_j^{Re}\right)$	0	0	x	y	z
$s_j = \left(s_j^{Se} \times s_k^{Re}\right)$	0	1	x	y	z
$s_k = \left(s_k^{Se} \times s_l^{Re}\right)$	1	0	x	y	z
$s_l = \left(s_l^{Se} \times s_i^{Re}\right)$	1	1	x	y	z
Trong đó: $x, y, z \in \{0,1\}$ và $x \neq y$ và $y \neq z$; $i, j, k, l = [\overline{0,3}] \subset \mathbb{N}^* = \mathbb{N} \cup (0)$					

(a) Ma trận này biểu diễn ba mươi hai trạng thái của năm thanh ghi



(b) Sơ đồ biểu diễn máy trạng thái đại số các RTL của ba mươi hai trạng thái

Hình 4.17. Kết nối các tiến trình của Se và Re và tích máy trạng thái đại số

4.7. Thuật toán 4.4 cho các kết quả tổng hợp RTL

4.7.1. Đặc tả thuật toán dựa trên ngôn ngữ mức cao

Thuật toán 4.3 dựa trên ngôn ngữ mức cao được mô tả, vì vậy một chương trình được tạo ra trong bước này. Công việc chính trong quá trình này là cập nhật các chế độ của các lập lịch khác nhau. Do đó, trong minh họa ngôn ngữ mô tả mức cao như được chỉ ra ở thuật toán 4.4 trình bày với mười ba sự kiện liên kết với các câu lệnh của thuật toán này, trong đó mỗi câu lệnh được xem như là một sự kiện.

4.7.2. Tạo mô hình dựa trên ngữ nghĩa đại số (ASM)

Vấn đề cốt lõi là việc tạo ra một đặc tả thứ tự cục bộ của các câu lệnh chương trình được tạo ra trong tiểu mục 4.6.1. Nói một cách khác, đặc tả này được sử dụng để thể hiện sự phụ thuộc dữ liệu của các câu lệnh cần thiết để tạo ra đặc tả hợp lý dựa trên ngữ nghĩa đại số của máy trạng thái được gọi tắt là ASA_{SPEC}. Để thực hiện nhiệm vụ này, cần tham chiếu đến một số mối quan hệ cơ bản giữa các sự kiện ở tiểu mục 4.3 gồm: *mối quan hệ độc lập, mối quan hệ trước sau, mối quan hệ xung đột,* và *mối quan hệ khả năng kích hoạt phân biệt.*

Tiếp cận của chương này giúp tạo ra mô hình dựa trên ngữ nghĩa đại số ASM_{SPEC} và sử dụng các mối quan hệ cơ bản giữa các sự kiện ở tiểu mục 4.3 để áp dụng vào trong ngữ cảnh của thuật toán 4.4. Những mối quan hệ này có thể được trích xuất từ

đặc tả hệ thống được cung cấp trong một ngôn ngữ mức cao như trong tiểu mục 4.6.1. Tập hợp sự kiện được ký hiệu là A và các mối quan hệ r giữa các sự kiện này là các thành phần chính của thuật toán. Những mối quan hệ này được trích xuất trực tiếp từ đặc tả của thuật toán. Tiếp cận này cho phép hiểu hơn về hành vi của thuật toán. Đây là một kỹ thuật có thể cung cấp thông tin quan trọng về cách hoạt động bên trong của thuật toán phức tạp.

Thuật toán 4.4 được phân bổ các sự kiện, gồm có mười ba sự kiện được ký hiệu là $e_0, e_1, e_2, e_3, \ldots, e_{12}$, ở đây e_0 xảy ra trước e_1, e_0 xảy ra trước e_2, e_1 độc lập với e_2, e_2 xảy ra trước e_3, e_0 xảy ra trước e_4, e_2 xảy ra trước e_4, e_3 độc lập với e_4, e_4 xảy ra trước e_5, e_5 độc lập với e_6, e_6 xảy ra trước e_7, e_7 xảy ra trước e_8, e_8 xảy ra trước e_9, e_6 xảy ra trước e_{10}, e_9 xảy ra trước e_{10}, e_{10} xảy ra trước e_{11}, e_7 xảy ra trước e_{11} và e_7 xảy ra trước e_{12}. Sự kết hợp của những mối quan hệ này cho chúng ta về mặt đặc tả ngữ nghĩa đại số cho thuật toán 4.4. Một cách hình thức, ASM_{SPEC} được định nghĩa hình thức bên dưới, từ danh sách các sự kiện của thuật toán 4.4 và các mối quan hệ giữa các sự kiện được tạo ra trong ASM_{SPEC}, một máy trạng thái đặc tả đại số của ASM_{SPEC} được tạo ra ở hình 4.18.

$$\mathbf{ASM_{SPEC}} = \{\boldsymbol{e_0 \prec e_1, e_0 \prec e_2, e_1 \nabla e_2, e_2 \prec e_3, e_0 \prec e_4, e_2 \prec e_4, e_3 \nabla e_4, e_4 \prec e_5, e_5 \nabla e_6, e_6 \prec e_7, e_7 \prec e_8, e_8 \prec e_9, e_6 \prec e_{10}, e_9 \prec e_{10}, e_{10} \prec e_{11}, e_7 \prec e_{11}, e_7 \prec e_{12}}\}$$

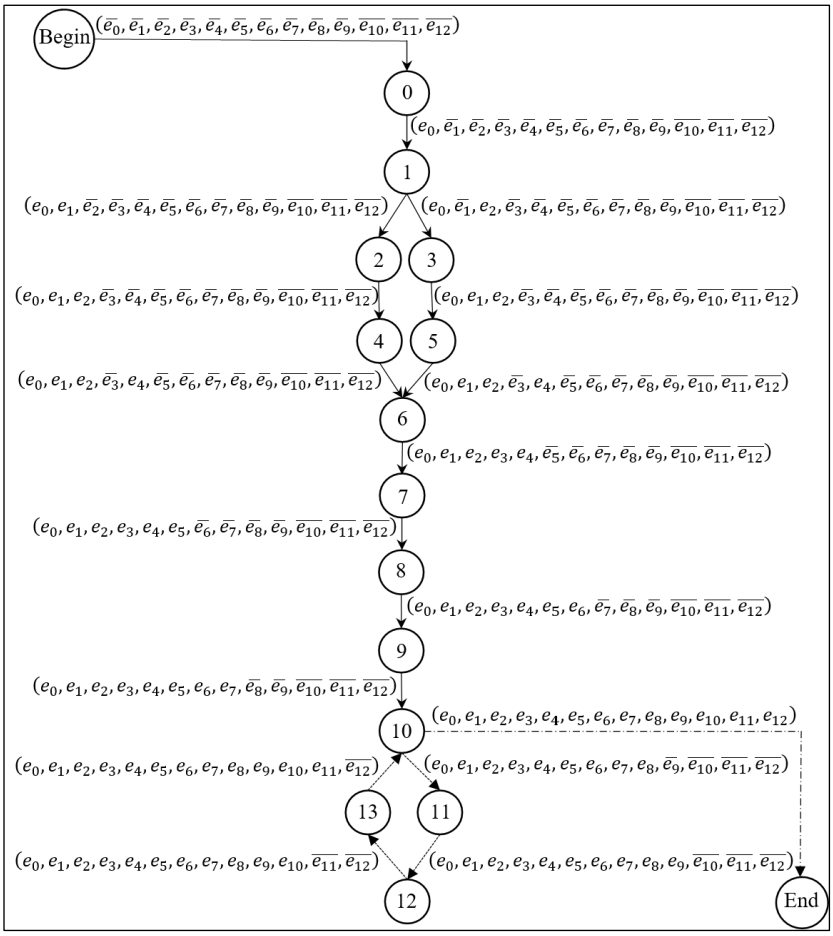
ASA_{SPEC}	e_0	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	e_{11}	e_{12}	Công thức logic (ASM_{SPEC})
s_0	1	1	1	1	1	1	1	1	1	1	1	0	0	$f = f_{e_0 \prec e_1} \wedge f_{e_0 \prec e_2} \wedge f_{e_1 \nabla e_2} \wedge f_{e_2 \prec e_3} \wedge f_{e_0 \prec e_4} \wedge f_{e_2 \prec e_4} \wedge f_{e_3 \nabla e_4} \wedge f_{e_4 \prec e_5} \wedge f_{e_5 \nabla e_6} \wedge f_{e_6 \prec e_7} \wedge f_{e_7 \prec e_8} \wedge f_{e_8 \prec e_9} \wedge f_{e_6 \prec e_{10}} \wedge f_{e_9 \prec e_{10}} \wedge f_{e_{10} \prec e_{11}} \wedge f_{e_7 \prec e_{11}} \wedge f_{e_7 \prec e_{12}} = (e_0 + \bar{e_1}) \wedge (e_0 + \bar{e_2}) \wedge 1 \wedge (e_2 + \bar{e_3}) \wedge (e_0 + \bar{e_4}) \wedge (e_2 + \bar{e_4}) \wedge 1 \wedge (e_4 + \bar{e_5}) \wedge 1 \wedge (e_6 + \bar{e_7}) \wedge (e_7 + \bar{e_8}) \wedge (e_8 + \bar{e_9}) \wedge (e_6 + \overline{e_{10}}) \wedge (e_9 + \overline{e_{10}}) \wedge (e_{10} + \overline{e_{11}}) \wedge (e_7 + \overline{e_{11}}) \wedge (e_7 + \overline{e_{12}}) = e_0 e_2 e_4 e_6 e_7 e_8 e_9 e_{10} + e_0 e_2 e_4 e_6 e_7 e_8 \overline{e_{10}} \overline{e_{11}} + e_0 e_2 e_4 e_6 e_7 \bar{e}_9 \overline{e_{10}} \overline{e_{11}} + e_0 e_2 e_4 e_6 \bar{e}_8 \bar{e}_9 \overline{e_{10}} \overline{e_{11}} \overline{e_{12}} + e_0 e_2 e_4 \overline{e_7} \bar{e}_8 \bar{e}_9 \overline{e_{10}} \overline{e_{11}} \overline{e_{12}} + e_0 e_2 \overline{e_5} \overline{e_7} \bar{e}_8 \bar{e}_9 \overline{e_{10}} \overline{e_{11}} \overline{e_{12}} + e_0 \bar{e}_3 \bar{e}_4 \overline{e_5} \overline{e_7} \bar{e}_8 \bar{e}_9 \overline{e_{10}} \overline{e_{11}} \overline{e_{12}} + \bar{e}_1 \bar{e}_2 \bar{e}_3 \bar{e}_4 \overline{e_5} \overline{e_7} \bar{e}_8 \bar{e}_9 \overline{e_{10}} \overline{e_{11}} \overline{e_{12}}$
s_1	1	1	1	1	1	1	1	1	1	1	1	1	0	
s_2	1	1	1	1	1	1	1	1	1	1	1	1	1	
s_3	1	1	1	1	1	1	1	1	1	0	0	0	0	
s_4	1	1	1	1	1	1	1	1	1	1	0	0	0	
s_5	1	1	1	1	1	1	1	1	0	0	0	0	0	
s_6	1	1	1	1	1	1	1	0	0	0	0	0	0	
s_7	1	1	1	0	1	0	0	0	0	0	0	0	0	
s_8	1	1	1	1	1	0	0	0	0	0	0	0	0	
s_9	1	1	1	1	1	1	0	0	0	0	0	0	0	
s_{10}	1	0	1	0	0	0	0	0	0	0	0	0	0	
s_{11}	1	1	1	0	0	0	0	0	0	0	0	0	0	
s_{12}	1	0	0	0	0	0	0	0	0	0	0	0	0	
s_{13}	1	1	0	0	0	0	0	0	0	0	0	0	0	
s_{14}	0	0	0	0	0	0	0	0	0	0	0	0	0	

Hình 4.18. ASA_{SPEC} của thuật toán 4.4

4.7.3. Kết quả tổng hợp RTL

Đây là kết quả của tổng hợp RTL cho một đặc tả hành vi thuật toán, một bước quan trọng trong thiết kế các thành phần sau đây [CT.2].

- *Đơn vị xử lý:* Chứa các khai báo của các thành phần tạo nên đơn vị xử lý. Nó chịu trách nhiệm thực thi các hoạt động được xác định trong mô tả RTL.
- *Đơn vị điều khiển:* Đơn vị này xác định chuỗi lệnh nội bộ phải được phát ra bởi đơn vị điều khiển. Nó trực tiếp điều hướng hoạt động của bộ xử lý và điều phối các hoạt động của các thành phần phần cứng khác.
- *Gán cố định:* Điều này xác định hoạt động phải được lặp lại trong mỗi xung đồng hồ. Đó là một phần quan trọng của thiết kế đảm bảo hoạt động chính xác của mạch.



Hình 4.19. Máy trạng thái dựa trên ngữ nghĩa đại số của thuật toán 4.4

Sau giai đoạn tổng hợp, thiết kế chuyển sang giai đoạn kiểm chứng, trong đó chức năng và hiệu năng của thiết kế được kiểm tra so với các đặc tả ban đầu. Đây là một bước quan trọng để đảm bảo tính đáng tin cậy và khả năng của sản phẩm cuối cùng. Đơn vị điều khiển được tạo thành từ các bước, mỗi bước được đánh số và phải được thực thi trong một đơn vị xung đồng hồ duy nhất. Thuật toán 4.4 cho thấy đơn vị điều khiển.

4.7.4. Tạo máy trạng thái RTL đại số

Máy trạng thái ASA_{RTL} của phần này cần được tạo ra từ kết quả tổng hợp RTL của thuật toán 4.4. Do đó, từ tổng hợp RTL của thuật toán 4.4 đã được chỉ ra ở hình 4.19 và máy trạng thái ASA_{RTL} đã được tạo ra ở hình 4.20.

ASA_{RTL}	e_0	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	e_{11}	e_{12}
s_0	0	0	0	0	0	0	0	0	0	0	0	0	0

ASA_{RTL}	e_0	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	e_{11}	e_{12}
s_1	1	0	0	0	0	0	0	0	0	0	0	0	0
s_2	1	0	1	0	0	0	0	0	0	0	0	0	0
s_3	1	1	0	0	0	0	0	0	0	0	0	0	0
s_4	1	1	1	0	0	0	0	0	0	0	0	0	0
s_5	1	1	1	0	1	0	0	0	0	0	0	0	0
s_6	1	1	1	1	1	0	0	0	0	0	0	0	0
s_7	1	1	1	1	1	1	0	0	0	0	0	0	0
s_8	1	1	1	1	1	1	1	0	0	0	0	0	0
s_9	1	1	1	1	1	1	1	1	0	0	0	0	0
s_{10}	1	1	1	1	1	1	1	1	1	0	0	0	0
s_{11}	1	1	1	1	1	1	1	1	1	1	0	0	0
s_{12}	1	1	1	1	1	1	1	1	1	1	1	0	0
s_{13}	1	1	1	1	1	1	1	1	1	1	1	1	0
s_{14}	1	1	1	1	1	1	1	1	1	1	1	1	1

Hình 4.20. ASA_{RTL} của thuật toán 4.4

4.8. Kết quả so sánh và thảo luận

4.8.1. Kết quả so sánh giữa ASM_{SPEC} và ASM_{RTL}

Trong việc so sánh giữa ASA_{SPEC} và ASA_{RTL} (của thuật toán 4.2 hoặc thuật toán 4.4) để kiểm chứng tính đúng đắn của các kết quả tổng hợp RTL và SPEC cần xác định các tính chất sau.

Định nghĩa 4.3 (Tính đúng đắn của kết quả tổng hợp RTL): Một kết quả tổng hợp RTL dựa trên ngữ nghĩa đại số là chính xác nếu đáp ứng tất cả các yêu cầu của đặc tả đại số.

Định lý 4.1: Một kết quả tổng hợp RTL dựa trên ngữ nghĩa đại số là chính xác nếu và chỉ nếu $\mathcal{C}(\text{RTL}) = \mathcal{C}(\text{SPEC})$ (Định nghĩa 4.1). Nói một cách khác, tập hợp các tính toán được chấp nhận bởi RTL tương đương với tập hợp tất cả các tính toán được chấp nhận bởi SPEC.

Chứng minh: Đối với phần "nếu", khi $\mathcal{C}(\text{RTL})$ tương đương với $\mathcal{C}(\text{SPEC})$, thì một kết quả tổng hợp RTL là chính xác theo định nghĩa 4.3. Để chứng minh phần "chỉ nếu", luận án cần chứng minh rằng nếu $\mathcal{C}(\text{RTL}) \neq \mathcal{C}(\text{SPEC})$ thì kết quả tổng hợp RTL không chính xác. Do đó, có hai trường hợp cần được phải kiểm chứng sau:

- Nếu $\mathcal{C}(\text{RTL}) \subset \mathcal{C}(\text{SPEC})$, thì tồn tại một yêu cầu tính toán $\mathcal{T} \in \mathcal{C}(\text{SPEC}) \setminus \mathcal{C}(\text{RTL})$ không được tổng hợp trong kết quả RTL. Theo định nghĩa 4.3, kết quả của tổng hợp RTL không chính xác.
- Nếu $\mathcal{C}(\text{SPEC}) \subset \mathcal{C}(\text{RTL})$, thì đặc tả và tổng hợp RTL không tương đương. Để rút ra hệ quả từ điều này, hành vi tính toán của chúng là khác nhau. Hay nói một cách khác, kết quả của tổng hợp RTL không chính xác.

4.8.2. Thảo luận

Những nhận xét trên cho thấy nếu thuật toán tổng hợp đã tạo ra một kết quả RTL hiệu quả và sau đó so sánh và đối chiếu được thực hiện, ở đây như là một đánh

giá để kiểm tra xem ASA_{RTL} là tương đương với ASA_{SPEC}. Nhận thấy, hình 4.13 và hình 4.15 của thuật toán 4.2; hình 4.18 và hình 4.20 của thuật toán 4.4, cả hai thuật toán này cho thấy $\mathcal{C}(\text{SPEC})$ là tương đương với $\mathcal{C}(\text{RTL})$. Do đó, tiếp cận kiểm chứng hình thức này đã chứng minh được tính đúng đắn của kết quả tổng hợp RTL cho thuật toán 4.2 và thuật toán 4.4 (Xem bảng 4.1).

Bảng 4.1. Nội dung so sánh ASA_{SPEC} và ASA_{RTL}

Nội dung so sánh	ASA_{SPEC}	ASA_{RTL}	Nhận xét / Kết luận
Nguồn gốc mô hình	Được đặc tả từ yêu cầu dựa theo không gian Chu	Được sinh ra từ thuật toán tổng hợp, tạo thành mô hình RTL thực thi.	ASA_{RTL} và ASA_{SPEC} là kết quả triển khai
Ngữ nghĩa hoạt động	Dựa trên đặc tả toán học ASA_{SPEC}	Dựa trên kết quả sinh ra từ thuật toán tổng hợp ASA_{RTL}	ASA_{RTL} tương đương với ASA_{SPEC}
Biểu diễn trong luận án	Hình 4.13, hình 4.15 của thuật toán 4.2.	Hình 4.18, hình 4.20 của thuật toán 4.4.	Hai cặp hình này minh họa sự tương đương hành vi thực thi và hành vi đặc tả của hai mô hình ASA_{RTL} và ASA_{SPEC}.
Hàm tính toán $\mathcal{C}(\text{SPEC})$ và $\mathcal{C}(\text{RTL})$	$\mathcal{C}(\text{SPEC})$: Hàm điều khiển trên ASA_{SPEC}	$\mathcal{C}(\text{RTL})$: Hàm tương ứng sinh ra trong ASA_{RTL}	$\mathcal{C}(\text{SPEC}) \equiv \mathcal{C}(\text{RTL})$
Tính đúng đắn	Là cơ sở để kiểm chứng đặc tả.	Kiểm chứng tính tương đương với đặc tả để đảm bảo RTL đúng.	Phương pháp kiểm chứng hình thức chứng minh tính đúng đắn của bước tổng hợp.
Kết luận chung	Đặc tả hành vi mong muốn.	Thực thi đúng đặc tả.	Các bước tổng hợp đã tạo một kết quả RTL cho thuật toán 4.2 (Phi vòng lặp) và thuật toán 4.4 (Vòng lặp) đều đúng đắn.

4.9. Phần kết chương 4

Chương 4 này đã tạo ra mô hình ngữ nghĩa đại số dựa trên không gian Chu để xử lý stream dạng BDL. Mô hình này cho phép đặc tả mối quan hệ giữa máy trạng thái mức chuyển đổi thanh ghi (ASA_{RTL}) và máy trạng thái đặc tả đại số (ASA_{SPEC}) của thuật toán 4.2 và thuật toán 4.4 để kiểm chứng các kết quả tổng hợp RTL và SPEC giúp luận án có được một trong ba kết luận sau đây:

- Một kết quả tổng hợp RTL và SPEC dựa trên ngữ nghĩa đại số là chính xác nếu và chỉ nếu $\mathcal{C}(\text{RTL}) = \mathcal{C}(\text{SPEC})$;
- Nếu $\mathcal{C}(\text{RTL}) \subset \mathcal{C}(\text{SPEC})$ thì tồn tại một tính toán $\mathcal{T} \in \mathcal{C}(\text{SPEC}) \setminus \mathcal{C}(\text{RTL})$ không được tổng hợp trong kết quả RTL dẫn đến kết quả của tổng hợp RTL không chính xác;

- Nếu $\mathcal{C}(\text{SPEC}) \subset \mathcal{C}(\text{RTL})$ thì đặc tả và tổng hợp RTL không tương đương suy ra hành vi tính toán của chúng là khác nhau.

Nhận thấy, để kiểm chứng tính đúng đắn của một kết quả tổng hợp RTL, luận án đã chứng minh rằng kết quả sử dụng một ASM là chính xác nếu và chỉ nếu $\mathcal{C}(\text{RTL}) = \mathcal{C}(\text{SPEC})$. Nói một cách khác, tập hợp tất cả các tính toán được đồng thuận bởi ASA_{RTL} tương đương với những điều được đồng thuận bởi ASA_{SPEC}.

KẾT LUẬN

❖ ĐÓNG GÓP KHOA HỌC

Luận án đã nghiên cứu, phân tích, phân loại, tổng hợp và so sánh các tiêu chí đặc trưng của lý thuyết stream, từ đó cho thấy rằng lý thuyết stream (bao gồm đại số stream và đồng đại số stream) là nền tảng lý thuyết cơ bản để xây dựng một số khung lý thuyết hình thức cho BDL và để phát triển các hệ thống BDL phổ biến hiện nay. Luận án đã mở rộng lý thuyết stream để giải thích, diễn giải và chứng minh cơ chế hoạt động của stream dạng BDL, điều mà trước đây chưa có một nền tảng toán học nào thực hiện. Các đóng góp khoa học của luận án như sau:

- **Đóng góp 1: Xây dựng cơ sở đại số stream dùng cho stream dạng BDL trong IoMT [CT.1][CT.2][CT.3][CT.5][CT.7][CT.8]**
  - *Phân tích lý thuyết stream:* Các nghiên cứu đã tiến hành phân tích toàn diện lý thuyết stream như là cơ sở cho các nghiên cứu sau này.
  - *Xây dựng mười phép toán xử lý stream:* Xây dựng một tập hợp mười phép toán xử lý stream dạng BDL, cung cấp các khả năng tương tác phong phú cho các hệ thống stream dạng BDL.
  - *Tổ hợp các phép toán xử lý stream thành các biểu thức stream mới:* Tách biệt, quan hệ, và kết hợp các phép toán xử lý stream dạng BDL thành những biểu thức stream mới phức tạp.
  - *Mở rộng đại số stream và đồng đại số stream:* Nghiên cứu và phát triển hình thái mở rộng của đại số stream và đồng đại số stream đối với stream dạng BDL trong CS.
  - *Cấu trúc đại số monoid:* Khảo sát và xây dựng cấu trúc đại số monoid như một khía cạnh đại số cơ bản.
  - *Tính chất monoid:* Phân tích các tính chất của monoid và ứng dụng vào stream trong hệ sinh thái IoMT.
  - Xây dựng kịch bản để mô phỏng và kiểm chứng đánh giá các phép toán xử lý stream dùng cho các CS.
- **Đóng góp 2: Xây dựng cơ chế lập lịch xử lý stream dạng BDL [CT.2][CT.9]**
  - *Phân hoạch các phép toán stream:* Tổ chức phân chia các phép toán theo các khối hợp lý.
  - *Lập lịch cho tính toán BDL:* Quá trình xác định và phân phối các bước tính toán cần thiết cho một ứng dụng cụ thể. Trong đó, một tiến trình được đặc tả bởi một số bước tính toán $k \geq 0$, tương ứng với các phép toán stream dạng BDL mà ứng dụng yêu cầu thực hiện. Mỗi phép toán xử lý stream này sẽ được

gán vào một CStep cụ thể, đảm bảo rằng trình tự thực thi được tối ưu hóa theo thời gian. Các CStep được đánh số theo thứ tự $0, 1, \ldots k$, thể hiện các bước tuần tự trong quá trình tính toán stream. Việc lập lịch là phân phối hợp lý các phép toán xử lý stream sao cho đáp ứng được các ràng buộc về hiệu năng, tối ưu hóa độ trễ, và đảm bảo tính đúng đắn của tiến trình thực thi.

- *Cấp phát và liên kết thanh ghi:* Thiết kế cơ chế quản lý thanh ghi hiệu quả.
- *Cấp phát và liên kết đơn vị chức năng:* Tối ưu hóa việc cấp phát đơn vị chức năng trong xử lý stream dạng BDL.
- *Tối ưu hóa lập lịch theo cơ chế đường ống:* Xây dựng mô hình quy hoạch tuyến tính để tối ưu hóa lập lịch theo cơ chế đường ống trong xử lý stream dạng BDL. Việc tối ưu hóa lập lịch này nhằm mang lại khai thác các lợi ích cốt lõi gồm: Giảm độ trễ, tăng thông lượng, cân bằng tải, và tối ưu hóa tài nguyên.

- **Đóng góp 3: Xây dựng mô hình ngữ nghĩa đại số (ASM) [CT.4][CT.6]**
  - *Mô hình ASM:* Xây dựng mô hình ngữ nghĩa đại số ASM nhằm xử lý stream dạng BDL một cách có hệ thống.
  - *Quan hệ giữa ASA_{RTL} và ASA_{SPEC}:* Đặc tả mối quan hệ giữa hai loại máy trạng thái ASA_{RTL} và ASA_{SPEC} để kiểm chứng các kết quả tổng hợp SPEC và RTL.

❖ HƯỚNG PHÁT TRIỂN

Các hướng phát triển của luận án không chỉ nhấn mạnh việc giải quyết các vấn đề kỹ thuật hiện tại mà còn mở ra các tiềm năng lớn cho việc phát triển lý thuyết và ứng dụng thực tế trong các hệ thống xử lý stream dạng BDL. Luận án đề xuất một số hướng phát triển trong tương lai như sau:

- **Xây dựng mô hình tổng quát tối ưu hóa lập lịch theo cơ chế đường ống**
  - Mô hình tổng quát tối ưu hóa lập lịch theo cơ chế đường ống.
  - Mô hình tối ưu hóa lập lịch theo cơ chế đường ống cho thuật toán vòng lặp.
- **Mở rộng các phép toán xử lý stream dạng BDL**
  - Phát triển thêm các phép toán stream vượt ngoài mười phép toán đã có nhằm nâng cao khả năng mô hình hóa và kiểm chứng hình thức cơ chế hoạt động cho stream dạng BDL.
  - Tìm kiếm các cách tiếp cận mới để hình thức hóa toàn diện stream dạng BDL thông qua việc mở rộng các phép toán xử lý stream.
  - Áp dụng các mô hình lý thuyết stream để xây dựng ứng dụng cho dữ liệu lớn dạng livestream trong thực tiễn.
- **Xây dựng tiến trình phân rã stream dạng BDL**

- Xây dựng và phát triển tiến trình phân rã stream, chia nhỏ stream dạng BDL thành các frame nhỏ hơn để áp dụng vào xử lý stream.
- Giải quyết các thách thức trong việc tối ưu hóa và triển khai hiệu quả tiến trình phân rã stream trong thực tiễn.

- **Xây dựng các cấu trúc đại số mới cho stream dạng BDL**
  - Mở rộng từ cấu trúc đại số monoid đã phát triển sang việc xây dựng các cấu trúc đại số phức tạp hơn như: dàn; vành; trường để hình thức hóa cơ chế hoạt động cho stream dạng BDL.
  - Đảm bảo các cấu trúc đại số này đáp ứng được yêu cầu mô hình hóa và kiểm chứng hình thức cho stream dạng BDL.
- **Phát triển lý thuyết stream tích hợp với dữ liệu lớn**
  - Phân tích và mở rộng lý thuyết stream để phục vụ xử lý dữ liệu lớn, với sự tích hợp của:
    - Các cấu trúc monoid liên quan.
    - Các mô hình biến đổi dữ liệu lớn.
    - Các máy trạng thái dùng cho việc chuyển đổi dữ liệu lớn.
    - Các bộ tổ hợp cho máy trạng thái trong ngữ cảnh dữ liệu lớn.
- **Nghiên cứu tính toán dữ liệu lớn trong hệ thống tính toán**
  - Tập trung vào các đặc tính cụ thể của hệ thống tính toán liên quan đến dữ liệu lớn như sau:
    - Các phép biến đổi dữ liệu.
    - Máy trạng thái và các bộ tổ hợp trạng thái.
  - Phát triển các phương pháp và công cụ hỗ trợ nghiên cứu, khai thác sâu hơn những đặc tính này trong tương lai.
- **Xây dựng một số khía cạnh ngữ nghĩa đại số trong hoạt động của BDL**
  - Phát triển một số tính chất ngữ nghĩa đại số trong hoạt động của BDL.
  - Phát triển một số luật dựa trên cấu trúc đại số trong hoạt động của BDL.
  - Phát triển một số bộ tổ hợp và các quan hệ của chúng trong hoạt động của BDL.

DANH MỤC CÔNG TRÌNH CÔNG BỐ CỦA LUẬN ÁN

[CT.1] **Dang Van Pham**, Vinh Cong Phan (2023), "Overview of The Stream Theory-Based Big Data in Livestream", Mobile Networks and Applications, ISSN: 1572-8153, Springer, Volume: 29, Issue: 4, pp: 1239–1256, DOI: 10.1007/s11036-023-02180-0, **Scopus indexed (Q1), SCIE IF: 3.8 (Q2)**.

[CT.2] **Dang Van Pham**, Vinh Cong Phan, Bao Khang Nguyen (2023), "Formally Specifying and Coinductive Approach to Verifying Synthesis of Stream Calculus-Based Computing Big Data in Livestream", Internet of Things, ISSN: 2542-6605, Elsevier, Volume: 23, DOI: 10.1016/j.iot.2023.100878, **Scopus indexed (Q1), SCIE IF: 7.6 (Q1)**.

[CT.3] **Dang Van Pham**, Vinh Cong Phan (2024), "Algebraic Aspects of Big Data in Livestream in Internet of Mobile Things", Mobile Networks and Applications, ISSN: 1572-8153, Springer, Volume: 29, Issue: 1, pp: 243–261, DOI: 10.1007/s11036-024-02297-w, **Scopus indexed (Q1), SCIE IF: 3.8 (Q2)**.

[CT.4] **Dang Van Pham**, Vinh Cong Phan, Bao Khang Nguyen (2024), "Algebraic Semantics of Register Transfer Level in Synthesis of Stream Calculus-Based Computing Big Data in Livestream", EAI ICTCC 2023 - 9th EAI International Conference on Nature of Computation and Communication. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering - LNICST, ISSN: 1867-8211, Springer, Cham, Volume: 586, pp: 19-35, ISBN: 978-3-031-59461-8, DOI: 10.1007/978-3-031-59462-5_2 **(Scopus, Q4)**.

[CT.5] **Dang Van Pham**, Vinh Cong Phan (2024), "Monoids Structure used for Computing Big Data in Livestream in Computing Systems", Hội thảo quốc gia lần thứ XXVII: Một số vấn đề chọn lọc của Công nghệ thông tin và truyền thông – VNICT 2024, Nha Trang, Việt Nam, 11-12/10/2024, trang: 363-369, ISBN: 978-604-67-3029-3.

[CT.6] **Dang Van Pham**, Vinh Cong Phan, Trong Toan Tran (2025), "Relationship between RTL Automata (ASA_{RTL}) and Algorithm Specification (ASA_{SPEC}) Based on Algebraic Semantics in Synthesis of Stream Calculus-Based Computing Big Data in Livestream", EAI ICCASA 2024 - 13th EAI International Conference on Context-Aware Systems and Applications. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering - LNICST, ISSN: 1867-8211, Springer, Cham, Volume: 664, ISBN: 978-3-032-12941-3 **(Scopus, Q4)**.

[CT.7] **Dang Van Pham**, Vinh Cong Phan, Toan Trong Tran, Bao Khang Nguyen (2024), "An Algebraic Structure for Computing Big Data in Livestream", 2024 13th International Conference on Control, Automation and Information Sciences (ICCAIS), ISSN: 2475-7896, ISBN: 979-8-3315-4204-7, IEEE, DOI: 10.1109/ICCAIS63750.2024.10814295 **(Scopus)**.

[CT.8] **Dang Van Pham**, Vinh Cong Phan, Trong Toan Tran (2025), "Analyzing and Comparing the Stream Theory: A Review", EAI ICTCC 2024 - 10th EAI International Conference on Nature of Computation and Communication. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering - LNICST, ISSN: 1867-8211, Springer, Cham, Volume: 668, ISBN: 978-3-032-12846-1, DOI: 10.1007/978-3-032-12846-1_6 **(Scopus, Q4)**.

[CT.9] **Dang Van Pham**, Vinh Cong Phan (2025), "Developing Linear Programming-Based Pipeline Scheduling Optimization Model Used to Big Data in Livestream", Proceeding of the 18th National Conference on Fundamental and Applied IT Research – FAIR'18, Ha Noi, 21-22/08/2025, ISBN: 978-604-357-455-5, DOI: 10.15625/vap.2025.0298, pp: 275-281 **(HĐCDGSNN)**.

DANH MỤC TÀI LIỆU THAM KHẢO

[1] B. Tidke and R. Mehta, "A Comprehensive Review and Open Challenges of Stream Big Data," in *Soft Computing: Theories and Applications*, Singapore, M. Pant, K. Ray, T. K. Sharma, S. Rawat, and A. Bandyopadhyay, Eds., 2018, vol. 584: Springer Singapore, pp. 89–99, ISBN: 978–981–10–5699–4, doi: 10.1007/978-981-10-5699-4_10.

[2] A. M. S. Osman, "A novel big data analytics framework for smart cities," *Future Generation Computer Systems*, vol. 91, pp. 620–633, 2019, doi: 10.1016/j.future.2018.06.046.

[3] X. Liu and S. H. Kim, "Beyond Shopping: The Motivations and Experience of Live Stream Shopping Viewers," in *2021 13th International Conference on Quality of Multimedia Experience (QoMEX)*, 2021, pp. 187–192, doi: 10.1109/QoMEX51781.2021.9465387.

[4] JianQin Zhao and H. Li, "The rising of livestream business model: Insights fromthe case study of TikTok in the UK," *The 22nd International Conference on Electronic Business, Bangkok, Thailand*, vol. 22, pp. 535–543, 2022.

[5] S. Zheng, M. Wu, and J. Liao, "The impact of destination live streaming on viewers' travel intention," *Current Issues in Tourism*, pp. 1–15, 2022, doi: 10.1080/13683500.2022.2117594.

[6] C. Xie, J. Yu, S. Huang, and J. Zhang, "Tourism e-commerce live streaming: Identifying and testing a value-based marketing framework from the live streamer perspective," *Tourism Management*, vol. 91, p. 104513, 2022, doi: 10.1016/j.tourman.2022.104513.

[7] S. M. van Bonn, J. S. Grajek, A. Schneider, T. Oberhoffner, R. Mlynski, and N. M. Weiss, "Interactive live-stream surgery contributes to surgical education in the context of contact restrictions," *Eur Arch Otorhinolaryngol*, 2021, doi: 10.1007/s00405-021-06994-0.

[8] T. Kubica, T. Hara, I. Braun, and A. Schill, "An Approach to Support Interactive Activities in Live Stream Lectures," in *Addressing Global Challenges and Quality Education*, Cham, 2020, vol. 12315: Springer International Publishing, pp. 432–436, doi: 10.1007/978-3-030-57717-9_40.

[9] J. Rot, F. Bonchi, M. Bonsangue, D. Pous, J. A. N. Rutten, and A. Silva, "Enhanced coalgebraic bisimulation," *Mathematical Structures in Computer Science*, vol. 27, no. 7, pp. 1236–1264, 2017, doi: 10.1017/S0960129515000523.

[10] H. H. Hansen, C. Kupke, and J. J. M. M. Rutten, "Stream Differential Equations: Specification Formats and Solution Methods," *Log. Methods Comput. Sci.*, vol. 13, no. 1, 2017, doi: 10.23638/LMCS-13(1:3)2017.

[11] M. Niqui and J. J. M. M. Rutten, "Stream processing coalgebraically," *Science of Computer Programming*, vol. 78, no. 11, pp. 2192–2215, 2013, doi: 10.1016/j.scico.2012.07.013.

[12] C. Kupke, M. Niqui, and J. Rutten, "Stream Differential Equations: concrete formats for coinductive definitions," Oxford University (2011), Tech. Report No.RR-11-10, 2011.

[13] R. Hinze, "Concrete stream calculus: An extended study," *Journal of Functional Programming*, vol. 20, no. 5-6, pp. 463–535, 2011, doi: 10.1017/S0956796810000213.

[14] S. Milius, "A Sound and Complete Calculus for Finite Stream Circuits," in *Proceedings of the Annual IEEE Symposium on Logic in Computer Science*, 2010: IEEE Computer Society, pp. 421–430, doi: 10.1109/lics.2010.11.

[15] J. Rutten, "Algebra, bitstreams, and circuits," CWI, Technical Report SEN-R0502, ISSN: 1386-369X, Jan 2005, vol. 16.

[16] J. J. M. M. Rutten, "An Application of Stream Calculus to Signal Flow Graphs," in *Formal Methods for Components and Objects*, Berlin, Heidelberg, F. S. de Boer, M. M. Bonsangue, S. Graf, and W.-P. de Roever, Eds., 2004: Springer Berlin Heidelberg, pp. 276–291, doi: 10.1007/978-3-540-30101-1_13.

[17] J. J. M. M. Rutten, "Elements of stream calculus (an extensive exercise in coinduction)," *Electronic Notes in Theoretical Computer Science*, vol. 45, pp. 358–423, 2001, doi: 10.1016/S1571-0661(04)80972-1. CWI (Centre for Mathematics and Computer Science), Published by Elsevier Science B. V.

[18] K. Terayama and H. Tsuiki, "A Stream Calculus of Bottomed Sequences for Real Number Computation," *Electronic Notes in Theoretical Computer Science*, vol. 298, pp. 383–402, 2013, doi: 10.1016/j.entcs.2013.09.023.

[19] M. Gaboardi and A. Saurin, "A foundational calculus for computing with streams," in *12th Italian Conference on Theoretical Computer Science*, 2010.

[20] J. Rutten, "The Method of Coalgebra: exercises in coinduction," CWI, Amsterdam, The Netherlands, ISBN: 978-90-6196-568-8, 2019, p. 261.

[21] M. Hirzel, G. Baudart, A. Bonifati, E. D. Valle, S. Sakr, and A. A. Vlachou, "Stream Processing Languages in the Big Data Era," *SIGMOD Rec.*, vol. 47, no. 2, pp. 29–40, 2018, doi: 10.1145/3299887.3299892.

[22] N. K. Doanh, L. Do Dinh, and N. N. Quynh, "Tea farmers' intention to participate in Livestream sales in Vietnam: The combination of the Technology Acceptance Model (TAM) and barrier factors," *Journal of Rural Studies*, vol. 94, pp. 408–417, 2022, doi: 10.1016/j.jrurstud.2022.05.023.

[23] M. Naeem *et al.*, "Trends and Future Perspective Challenges in Big Data," in *Advances in Intelligent Data Analysis and Applications*, Singapore, 2022: Springer Singapore, pp. 309–325, ISBN: 978–981–16–5036–9, doi: 10.1007/978-981-16-5036-9_30.

[24] A. Mohamed, M. K. Najafabadi, Y. B. Wah, E. A. K. Zaman, and R. Maskat, "The state of the art and taxonomy of big data analytics: view from new big data framework," *Artificial Intelligence Review*, vol. 53, no. 2, pp. 989–1037, 2020, doi: 10.1007/s10462-019-09685-9.

[25] V. E. Venugopal, M. Theobald, D. Tassetti, S. Chaychi, and A. Tawakuli, "Targeting a light-weight and multi-channel approach for distributed stream processing," *Journal of Parallel and Distributed Computing*, vol. 167, pp. 77–96, 2022, doi: 10.1016/j.jpdc.2022.04.022.

[26] M. Fragkoulis, P. Carbone, V. Kalavri, and A. Katsifodimos, "A survey on the evolution of stream processing systems," *The VLDB Journal*, vol. 33, no. 2, pp. 507–541, ISSN: 0949–877X, 2024, doi: 10.1007/s00778-023-00819-8.

[27] A. Ashabi, S. B. Sahibuddin, and M. S. Haghighi, "Big Data: Current Challenges and Future Scope," in *IEEE 10th Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, 2020, pp. 131–134, doi: 10.1109/ISCAIE47305.2020.9108826.

[28] B. Mohanta, S. Patnaik, and S. Patnaik, "Big Data for Modelling Interactive Systems in IoT," in *2018 2nd International Conference on Data Science and Business Analytics (ICDSBA)*, 2018, pp. 105–110, doi: 10.1109/ICDSBA.2018.00026.

[29] M. Younas, "Research challenges of big data," *Service Oriented Computing and Applications*, vol. 13, no. 2, pp. 105–107, 2019, doi: 10.1007/s11761-019-00265-x.

[30] A. Awad, R. Tommasini, S. Langhi, M. Kamel, E. Della Valle, and S. Sakr, "D2IA: User-defined interval analytics on distributed streams," *Information Systems*, vol. 104, p. 101679, 2022, doi: 10.1016/j.is.2020.101679.

[31] H. Nasiri, S. Nasehi, and M. Goudarzi, "Evaluation of distributed stream processing frameworks for IoT applications in Smart Cities," *Journal of Big Data*, vol. 6, no. 1, p. 52, 2019, doi: 10.1186/s40537-019-0215-2.

[32] A. Oussous, F.-Z. Benjelloun, A. Ait Lahcen, and S. Belfkih, "Big Data technologies: A survey," *Journal of King Saud University - Computer and Information Sciences*, vol. 30, no. 4, pp. 431–448, 2018, doi: 10.1016/j.jksuci.2017.06.001.

[33] M. A. Helala, F. Z. Qureshi, and K. Q. Pu, "A Stream Algebra for Performance Optimization of Large Scale Computer Vision Pipelines," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 2, pp. 905–923, 2022, doi: 10.1109/TPAMI.2020.3015867.

[34] A. Geroldinger and Q. Zhong, "Factorization theory in commutative monoids," *Semigroup Forum*, vol. 100, no. 1, pp. 22–51, 2020, doi: 10.1007/s00233-019-10079-0.

[35] Z. Yang and N. Wu, "Modular Models of Monoids with Operations," *Proc. ACM Program. Lang.*, vol. 7, no. ICFP, p. Article 208, 2023, doi: 10.1145/3607850.

[36] J. J. M. M. Rutten, "A coinductive calculus of streams," *Mathematical Structures in Computer Science*, vol. 15, no. 1, pp. 93 – 147, 2005, doi: 10.1017/S0960129504004517. Association for Computing Machinery, Cambridge University Press United States.

[37] J. J. M. M. Rutten, "Behavioural differential equations: a coinductive calculus of streams, automata, and power series," *Theoretical Computer Science*, vol. 308, no. 1, pp. 1–53, ISSN: 0304–3975, 2003, doi: 10.1016/S0304-3975(02)00895-2.

[38] J. Kim, "Coinductive Properties of Causal Maps," in *Algebraic Methodology and Software Technology*, Berlin, Heidelberg, J. Meseguer and G. Roşu, Eds., 2008: Springer Berlin Heidelberg, pp. 253–267, doi: 10.1007/978-3-540-79980-1_20.

[39] M. Boreale, L. Collodi, and D. Gorla, "Products, Polynomials and Differential Equations in the Stream Calculus," *ACM Trans. Comput. Logic*, vol. 25, no. 1, pp. 1–26, ISSN: 1529–3785, 2024, doi: 10.1145/3632747.

[40] Michele Boreale and D. Gorla, "Algebra and Coalgebra of Stream Products," in *In 32nd International Conference on Concurrency Theory (CONCUR 2021). Leibniz International Proceedings in Informatics (LIPIcs)*, 2021, vol. 203: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, pp. 19:1–19:17, doi: 10.4230/LIPIcs.CONCUR.2021.19.

[41] J. Winter, M. M. Bonsangue, and J. J. M. M. Rutten, "Context-free coalgebras," *Journal of Computer and System Sciences*, vol. 81, no. 5, pp. 911–939, ISSN: 0022–0000, 2015, doi: 10.1016/j.jcss.2014.12.004.

[42] C. Kupke and J. J. M. M. Rutten, "On the Final Coalgebra of Automatic Sequences," in *Logic and Program Semantics: Essays Dedicated to Dexter Kozen on the Occasion of His 60th Birthday*, R. L. Constable and A. Silva Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 149–164, ISBN: 978–3–642–29485–3, DOI: 10.1007/978–3–642–29485–3_10.

[43] Y. Zakowski, P. He, C.-K. Hur, and S. Zdancewic, "An equational theory for weak bisimulation via generalized parameterized coinduction," in *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, New Orleans, LA, USA, 2020: Association for Computing Machinery, pp. 71–84, ISBN: 9783642396335, doi: 10.1145/3372885.3373813.

[44] F. Bonchi, M. Bonsangue, M. Boreale, J. Rutten, and A. Silva, "A coalgebraic perspective on linear weighted automata," *Information and Computation*, vol. 211, pp. 77–105, ISSN: 0890–5401, 2012, doi: 10.1016/j.ic.2011.12.002.

[45] D. Sangiorgi and J. Rutten, "Advanced Topics in Bisimulation and Coinduction," Cambridge University Press, 40 W. 20 St. New York, NYUnited States, 2011, pp. 340, ISBN: 978–1–107–00497–9, DOI: book/10.5555/2103601.
[46] F. Vahid, "Digital Design with RTL Design, VHDL, and Verilog," John Wiley and Sons, 2010, pp. 247, ISBN: 978–0–470–53108–2.
[47] Y. Maruyama, "Chu duality theory and coalgebraic representation of quantum symmetries," *Journal of Pure and Applied Algebra*, vol. 226, no. 9, p. 106960, 2022, doi: 10.1016/j.jpaa.2021.106960.
[48] C. Fields and J. F. Glazebrook, "Information flow in context-dependent hierarchical Bayesian inference," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 34, no. 1, pp. 111–142, 2022, doi: 10.1080/0952813X.2020.1836034.
[49] K. Buchenrieder, "Rapid Prototyping of Embedded Hardware/Software Systems," *Design Automation for Embedded Systems*, vol. 5, no. 3, pp. 215–221, 2000, doi: 10.1023/A:1008950900254.
[50] I. Ulidowski, I. Phillips, and S. Yuen, "Reversing Event Structures," *New Generation Computing*, vol. 36, no. 3, pp. 281–306, 2018, doi: 10.1007/s00354-018-0040-8.
[51] T. Akidau *et al.*, "The dataflow model: a practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing," *Proc. VLDB Endow.*, vol. 8, no. 12, pp. 1792–1803, 2015, doi: 10.14778/2824032.2824076.
[52] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource Management with Deep Reinforcement Learning," presented at the Proceedings of the 15th ACM Workshop on Hot Topics in Networks, Atlanta, GA, USA, 2016. [Online]. Available: DOI: 10.1145/3005745.3005750.
[53] G. Zhou, W. Tian, R. Buyya, R. Xue, and L. Song, "Deep reinforcement learning-based methods for resource scheduling in cloud computing: a review and future directions," *Artificial Intelligence Review*, vol. 57, no. 5, 2024, doi: 10.1007/s10462-024-10756-9.
[54] D. Cacciarelli and M. Kulahci, "Active learning for data streams: a survey," *Machine Learning*, vol. 113, no. 1, pp. 185–239, 2023, doi: 10.1007/s10994-023-06454-2.
[55] N. Tantalaki, S. Souravlas, M. Roumeliotis, and S. Katsavounis, "Pipeline-Based Linear Scheduling of Big Data Streams in the Cloud," *IEEE Access*, vol. 8, pp. 117182–117202, 2020, doi: 10.1109/ACCESS.2020.3004612.
[56] A. Shukla and Y. Simmhan, "Model-driven scheduling for distributed stream processing systems," *Journal of Parallel and Distributed Computing*, vol. 117, pp. 98–114, 2018, doi: 10.1016/j.jpdc.2018.02.003.
[57] S. Kovach, P. Kolichala, T. Gu, and F. Kjolstad, "Indexed Streams: A Formal Intermediate Representation for Fused Contraction Programs," *Proceedings of the ACM on Programming Languages*, vol. 7, no. PLDI, pp. 1169–1193, 2023, doi: 10.1145/3591268.
[58] M. Toro, M. Desainte-Catherine, D. Janin, and Y. Orlarey, "Real-time interactive streams and temporal objects language," *Osfpreprints*, 2018, doi: 10.31219/osf.io/a6stk.
[59] G. Bacci and M. Miculan, "Structural operational semantics for continuous state stochastic transition systems," *Journal of Computer and System Sciences*, vol. 81, no. 5, pp. 834–858, 2015, doi: 10.1016/j.jcss.2014.12.003.
[60] F. Bonchi, M. M. Bonsangue, H. H. Hansen, P. Panangaden, J. J. M. M. Rutten, and A. Silva, "Algebra-coalgebra duality in brzozowski's minimization algorithm," *ACM Trans. Comput. Logic*, vol. 15, no. 1, pp. 1–29, 2014, doi: 10.1145/2490818.
[61] M. Niqui and J. Rutten, "Sampling, Splitting and Merging in Coinductive Stream Calculus," in *Mathematics of Program Construction*, Berlin, Heidelberg, C. Bolduc, J.

Desharnais, and B. Ktari, Eds., 2010, vol. 6120: Springer Berlin Heidelberg, pp. 310–330, doi: 10.1007/978-3-642-13321-3_18.

[62] P. C. Vinh and J. P. Bowen, "Formalization of Data Flow Computing and a Coinductive Approach to Verifying Flowware Synthesis," *Lecture Notes in Computer Science* vol. 4750, pp. 1–36, 2008, doi: 10.1007/978-3-540-79299-4_1. Springer Berlin Heidelberg.

[63] J. J. M. M. Rutten, "A tutorial on coinductive stream calculus and signal flow graphs," *Theoretical Computer Science,* vol. 343, no. 3, pp. 443–481, 2005, doi: 10.1016/j.tcs.2005.06.019.

[64] F. Frank, S. Milius, and H. Urbat, "Coalgebraic Semantics for Nominal Automata," in *Coalgebraic Methods in Computer Science*, Cham, H. H. Hansen and F. Zanasi, Eds., 2022: Springer International Publishing, pp. 45–66, doi: 10.1007/978-3-031-10736-8_3.

[65] Henning Basold, Helle Hvid Hansen, Jean-Eric Pin, and J. Rutten, "Newton Series, Coinductively: A Comparative Study of Composition," in *Mathematical Structures in Computer Science*, 2019, pp. 1–29, doi: 10.1017/S0960129517000159.

[66] F. Bonchi, M. D. Lee, and J. Rot, "Bisimilarity of Open Terms in Stream GSOS," in *Fundamentals of Software Engineering*, Cham, M. Dastani and M. Sirjani, Eds., 2017: Springer International Publishing, pp. 35–50, ISBN: 978–3–319–68972–2, doi: 10.1007/978-3-319-68972-2_3.

[67] J. Rot, M. Bonsangue, and J. Rutten, "Proving language inclusion and equivalence by coinduction," *Information and Computation,* vol. 246, pp. 62–76, ISSN: 0890–5401, 2016, doi: 10.1016/j.ic.2015.11.009.

[68] H. H. Hansen, C. Kupke, J. Rutten, and J. Winter, "A Final Coalgebra for k-regular Sequences," in *Horizons of the Mind. A Tribute to Prakash Panangaden: Essays Dedicated to Prakash Panangaden on the Occasion of His 60th Birthday*, F. van Breugel, E. Kashefi, C. Palamidessi, and J. Rutten Eds. Cham: Springer International Publishing, 2014, pp. 363–383, ISBN: 978–3–319–06880–0, DOI: 10.1007/978–3–319–06880–0_19.

[69] J. Rot, M. Bonsangue, and J. Rutten, "Coalgebraic Bisimulation-Up-To," in *SOFSEM 2013: Theory and Practice of Computer Science*, Berlin, Heidelberg, P. van Emde Boas, F. C. A. Groen, G. F. Italiano, J. Nawrocki, and H. Sack, Eds., 2013, vol. 7741: Springer Berlin Heidelberg, pp. 369–381, ISBN: 978–3–642–35843–2, doi: 10.1007/978-3-642-35843-2_32.

[70] V. Capretta, "Coalgebras in functional programming and type theory," *Theoretical Computer Science,* vol. 412, no. 38, pp. 5006–5024, 2011, doi: 10.1016/j.tcs.2011.04.024.

[71] A. Silva and J. Rutten, "A coinductive calculus of binary trees," *Information and Computation,* vol. 208, no. 5, pp. 578–593, 2010, doi: 10.1016/j.ic.2008.08.006.

[72] M. Bonsangue, J. Rutten, and A. Silva, "An Algebra for Kripke Polynomial Coalgebras," in *2009 24th Annual IEEE Symposium on Logic In Computer Science*, 2009, pp. 49–58, doi: 10.1109/LICS.2009.18.

[73] J. Rutten, "Rational streams coalgebraically," *Logical Methods in Computer Science,* vol. 4, no. 3:9, pp. 1–22, 2008, doi: 10.2168/LMCS-4 (3:9) 2008.

[74] J. J. M. M. Rutten, "Algebraic Specification and Coalgebraic Synthesis of Mealy Automata," *Electronic Notes in Theoretical Computer Science,* vol. 160, pp. 305–319, 2006, doi: 10.1016/j.entcs.2006.05.030.

[75] M. M. Rathore, H. Son, A. Ahmad, A. Paul, and G. Jeon, "Real-Time Big Data Stream Processing Using GPU with Spark Over Hadoop Ecosystem," *International Journal of Parallel Programming,* vol. 46, no. 3, pp. 630–646, 2018, doi: 10.1007/s10766-017-0513-2.

[76] A. F. Ahmad, M. S. Sayeed, C. P. Tan, K. G. Tan, M. A. Bari, and F. Hossain, "A Review on IoT with Big Data Analytics," in *9th International Conference on Information and Communication Technology*, 2021, pp. 160–164, doi: 10.1109/ICoICT52021.2021.9527503.

[77] M. H. ur Rehman, I. Yaqoob, K. Salah, M. Imran, P. P. Jayaraman, and C. Perera, "The role of big data analytics in industrial Internet of Things," *Future Generation Computer Systems*, vol. 99, pp. 247–259, 2019, doi: 10.1016/j.future.2019.04.020.

[78] S. Sarker, K. Roy, F. Afroz, and A.-S. K. Pathan, "On the Opportunities, Applications, and Challenges of Internet of Things," in *Decentralised Internet of Things: A Blockchain Perspective*, M. A. Khan, M. T. Quasim, F. Algarni, and A. Alharthi Eds. Cham: Springer International Publishing, 2020, pp. 231–254.

[79] M. Ge, H. Bangui, and B. Buhnova, "Big Data for Internet of Things: A Survey," *Future Generation Computer Systems*, vol. 87, pp. 601–614, 2018, doi: 10.1016/j.future.2018.04.053.

[80] K. Nahrstedt, H. Li, P. Nguyen, S. Chang, and L. Vu, "Internet of Mobile Things: Mobility-Driven Challenges, Designs and Implementations," in *2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI)*, 2016, pp. 25–36, doi: 10.1109/IoTDI.2015.41.

[81] I. Tcarenko *et al.*, "Smart energy efficient gateway for Internet of mobile things," in *2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, 2017, pp. 1016–1017, doi: 10.1109/CCNC.2017.7983276.

[82] S. C. Tolem, C. R. Bogadi, N. S. Korlapati, S. Ravichandran, R. Rajendran, and C. Vuppalapati, "A Theoretical Study on Advances in Streaming Analytics," in *IEEE Sixth International Conference on Big Data Computing Service and Applications (BigDataService)*, 2020, pp. 41–45, doi: 10.1109/BigDataService49289.2020.00014.

[83] T. Zheng, G. Chen, X. Wang, C. Chen, X. Wang, and S. Luo, "Real-time intelligent big data processing: technology, platform, and applications," *Science China Information Sciences*, vol. 62, no. 8, p. 82101, 2019, doi: 10.1007/s11432-018-9834-8.

[84] M. Mortazavi-Dehkordi and K. Zamanifar, "Efficient deadline-aware scheduling for the analysis of Big Data streams in public Cloud," *Cluster Computing*, vol. 23, no. 1, pp. 241–263, 2020, doi: 10.1007/s10586-019-02908-2.

[85] T. Kolajo, O. Daramola, and A. Adebiyi, "Big data stream analysis: a systematic literature review," *Journal of Big Data*, vol. 6, no. 1, 2019, doi: 10.1186/s40537-019-0210-7.

[86] F. Kordon, L. M. Hillah, F. Hulin-Hubard, L. Jezequel, and E. Paviot-Adet, "Study of the efficiency of model checking techniques using results of the MCC from 2015 To 2019," *International Journal on Software Tools for Technology Transfer*, vol. 23, no. 6, pp. 931–952, 2021, doi: 10.1007/s10009-021-00615-1.

[87] A. Souri and M. Norouzi, "A State-of-the-Art Survey on Formal Verification of the Internet of Things Applications," *Journal of Service Science Research*, vol. 11, no. 1, pp. 47–67, 2019, doi: 10.1007/s12927-019-0003-8.

[88] R. Alur, "Formal verification of hybrid systems," presented at the Proceedings of the ninth ACM international conference on Embedded software, Taipei, Taiwan, 2011. [Online]. Available: 10.1145/2038642.2038685.

[89] V. T. Anh, P. C. Vinh, and P. Q. Cuong, "Contextual Perception in Internet of Mobile Things: A categorical structure," *Internet of Things*, vol. 22, p. 100799, 2023, doi: 10.1016/j.iot.2023.100799.

[90] A. C. Keizer, H. Basold, and J. A. Pérez, "Session Coalgebras: A Coalgebraic View on Regular and Context-free Session Types," *ACM Trans. Program. Lang. Syst.*, vol. 44, no. 3, 2022, doi: 10.1145/3527633.

[91] M. Girish, G. Gopakumar, and D. S. Divya, "Formal and Simulation Verification: Comparing and Contrasting the two Verification Approaches," in *2021 2nd International Conference on Advances in Computing, Communication, Embedded and Secure Systems (ACCESS)*, 2021, pp. 41–44, doi: 10.1109/ACCESS51619.2021.9563305.

[92] M. Nakamura, S. Higashi, K. Sakakibara, and O. a, "Formal verification of Fischer's real-time mutual exclusion protocol by the OTS/CafeOBJ method," in *2020 59th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, 2020, pp. 1210–1215, doi: 10.23919/SICE48898.2020.9240272.

[93] B. S. Neysian and S. M. Babamir, "Automatic verification of uml state chart by bogor model checking tool: Automatic formal verification of network and distributed systems," in *2015 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI)*, 2015, pp. 797–802, doi: 10.1109/KBEI.2015.7436146.

[94] M. Kezadri Hamiaz, M. Pantel, B. Combemale, and X. Thirioux, "A Formal Framework to Prove the Correctness of Model Driven Engineering Composition Operators," in *Formal Methods and Software Engineering*, Cham, S. Merz and J. Pang, Eds., 2014: Springer International Publishing, pp. 235–250, doi: 10.1007/978-3-319-11737-9_16.

[95] D. Bošnački and A. Wijs, "Model checking: recent improvements and applications," *International Journal on Software Tools for Technology Transfer*, vol. 20, no. 5, pp. 493–497, 2018, doi: 10.1007/s10009-018-0501-x.

[96] X. Shu, N. Luo, B. Wang, X. Wang, and L. Zhao, "Model Checking Java Programs with MSVL," in *Structured Object-Oriented Formal Language and Method*, Cham, Z. Duan, S. Liu, C. Tian, and F. Nagoya, Eds., 2019: Springer International Publishing, pp. 89–107, doi: 10.1007/978-3-030-13651-2_6.

[97] Manfred Droste and G.-Q. Zhang, "Bifinite Chu Spaces," *EPI Sciences Overlay Journal, lmcs:1183 - Logical Methods in Computer Science*, vol. 6, no. 1, 2010, doi: 10.2168/LMCS-6(1:3)2010.