TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI

PHÂN HIỆU TẠI TP. HỒ CHÍ MINH

BỘ MÔN CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN TỐT NGHIỆP

ĐỀ TÀI: NGHIÊN CỨU MÔ HÌNH COKB VÀ XÂY DỰNG HỆ THỐNG GIẢI TOÁN RỜI RẠC

Giảng viên hướng dẫn: TS. NGUYỄN ĐÌNH HIỂN

Sinh viên thực hiện: NGUYỄN PHÚC HOÀI LINH

Mã sinh viên: 5851071042

Lớp : CÔNG NGHỆ THÔNG TIN

Khoá : K58

Tp. Hồ Chí Minh, năm 2021

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI

PHÂN HIỆU TẠI TP. HỒ CHÍ MINH

BỘ MÔN CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN TỐT NGHIỆP

ĐỀ TÀI: NGHIÊN CỨU MÔ HÌNH COKB VÀ XÂY DỰNG HỆ THỐNG GIẢI TOÁN RỜI RẠC

Giảng viên hướng dẫn: TS. NGUYỄN ĐÌNH HIỂN

Sinh viên thực hiện: NGUYỄN PHÚC HOÀI LINH

Mã sinh viên: 5851071042

Lớp : CÔNG NGHỆ THÔNG TIN

Khoá : K58

Tp. Hồ Chí Minh, năm 2021

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
PHÂN HIỆU TẠI THÀNH PHỐ HỒ CHÍ MINH

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh phúc

NHIỆM VỤ THIẾT KẾ TỐT NGHIỆP

BỘ MÔN: **CÔNG NGHỆ THÔNG TIN**

-------***-------

Mã sinh viên:5851071042..... **Họ tên SV:** Nguyễn Phúc Hoài Linh.....
Khóa: K58..... **Lớp:** Công Nghệ Thông Tin.....

1. **Tên đề tài:** Nghiên cứu mô hình COKB và xây dựng hệ thống giải toán rời rạc
2. **Mục đích, yêu cầu:**
   - Nghiêng cứu các mô hình biểu diễn các miền tri thức Toán rời rạc dựa trên mô hình COKB.
   - Thiết kế các thuật giải để giải quyết các dạng bài tập trong các miền tri thức Toán rời rạc.
   - Xây dựng website hỗ trợ việc giải các bài toán trong Toán rời rạc theo tiêu chí từng bước, dễ hiểu và bám sát giáo trình.

3. Nội dung và phạm vi đề tài: Gồm hai giai đoạn: nghiên cứu và ứng dụng xây dựng mô hình cho miền tri thức toán rời rạc. Hai là ứng dụng các mô hình tìm được để phát triển một website. Phạm vi đề tài xoay quanh ba chương trong toán rời rạc là: Logic mệnh đề, đại số Boolean và quan hệ hai ngôi.

4. Công nghệ, công cụ và ngôn ngữ lập trình: Nodejs, Typescript và Javascript. Deploy trên Heroku.

5. **Các kết quả chính dự kiến sẽ đạt được và ứng dụng:** Xây dựng được một website giải được các bài toán trong các chương Logic mệnh đề, đại số Boolean và quan hệ hai ngôi.

6. Giảng viên và cán bộ hướng dẫn

Họ tên: TS. Nguyễn Đình Hiển
Đơn vị công tác: Trường Đại học Công nghệ thông tin, ĐHQG-HCM
Điện thoại: 0918735299 Email: hiennd@uit.edu.vn

Ngày tháng 03 năm 2021 **Đã giao nhiệm vụ TKTN**

Trưởng BM Công nghệ Thông tin **Giảng viên hướng dẫn**

ThS. Trần Phong Nhã

Đã nhận nhiệm vụ TKTN

Sinh viên: Nguyễn Phúc Hoài Linh Ký tên:

Điện thoại: 0335849481 Email: 5851071042@st.utc2.edu.vn

LỜI CẢM ƠN

Trong suốt khoảng thời gian hoàn thành đồ án tốt nghiệp này, em đã gặp không ít khó khăn và thử thách. Nhưng với sự giúp đỡ của các thầy cô, bạn bè và những người xung quanh đã giúp em hoàn thành được đồ án này.

Lời cảm ơn đầu tiên, em xin cảm ơn thầy TS. Nguyễn Đình Hiển, người đã bỏ thời gian và công sức đã chỉ dạy em hoàn thành khóa luận này. Em cũng chân thành cảm ơn, quý thầy cô giảng viên trường Đại học Giao Thông Vận Tải phân hiệu tại TPHCM nói chung và các thầy cô Bộ môn CNTT nói riêng, đã truyền đạt cho em những kiến thức để em có nền tảng như hiện nay.

Chân thành gửi lời cảm ơn đến các bạn trong lớp CNTT K58, những người luôn hỗ trợ hết mình cho các thành viên trong lớp mỗi khi ai đó gặp khó khăn. Luôn luôn chia sẻ các kiến thức để tập thể cùng phát triển.

Cuối cùng em xin cảm ơn những người thân đã luôn tạo điều kiện tốt nhất để em có thể hoàn thành khóa luận này, nhất là trong thời buổi dịch bệnh như ngày nay. Tất nhiên, bài báo cáo sẽ có nhiều thiếu sót, để có thể khắc phục những điều đó, em hi vọng rằng sẽ được sự giúp đỡ và góp ý chân thành từ phía các thầy, các cô.

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Tp. Hồ Chí Minh, ngày …… tháng …… năm ……

Giảng viên hướng dẫn

Nguyễn Đình Hiển

MỤC LỤC

MỤC LỤC i
DANH MỤC CHỮ VIẾT TẮT iv
BẢNG BIỂU, SƠ ĐỒ, HÌNH VẼ v
CHƯƠNG 1 MỞ ĐẦU 1
1.1. Giới thiệu 1
1.2. Mục tiêu nghiên cứu 2
1.3. Phạm vi 2
1.4. Cấu trúc báo cáo thực tập tốt nghiệp 2
CHƯƠNG 2 TỔNG QUAN 3
2.1. Các kiến thức miền tri thức toán rời rạc 3
2.1.1. Lý thuyết Logic mệnh đề 3
2.1.2. Lý thuyết về Hàm Boolean 5
2.1.3. Quan hệ hai ngôi. 6
2.2. Mô hình COKB 8
CHƯƠNG 3 XÂY DỰNG MÔ HÌNH CHO MIỀN TRI THỨC TOÁN RỜI RẠC 10
3.1. Xây dựng mô hình cho miền tri thức Logic mệnh đề 10
3.1.1. Cấu trúc (C,Ops,Rules) 10
3.1.2. Thành phần Funcs 13
3.2. Xây dựng mô hình cho miền tri thức đại số Boolean 14
3.2.1. Thành phần C 15
3.2.2. Thành phần Ops 16
3.2.3. Thành phần Rule: 16
3.2.4. Thành phần Funcs 18
3.3. Quan hệ hai ngôi 18
3.3.1. Thành phần $K_{SoNguyen}$ 19

3.3.2. Thành phần (C,H,Rules) 23
3.3.3. Thành phần Funcs. 25
CHƯƠNG 4 CÁC BÀI TOÁN Ở MIỀN TRI THỨC TOÁN RỜI RẠC **Error! Bookmark not defined.**
4.1. Mô hình cho cái bài toán logic mệnh đề 26
4.1.1. Sự tương đương logic 26
4.1.2. Sự suy diễn logic 29
4.1.3. Rút gọn mệnh đề 30
4.1.4. Chứng minh hai biểu thức tương đương 32
4.1.5. Suy diễn logic 33
4.1.6. Xác định chân trị biểu thức 36
4.2. Mô hình giải các bài toán đại số Boolean 37
4.2.1. Ánh xạ đa thức biểu diễn hàm Boolean lên bìa Karnaugh 37
4.2.2. Xác định tế bào lớn. 38
4.2.3. Xác định các nhóm tế bào lớn 40
4.2.4. Xác định đa thức tối giản. 40
4.3. Xây dựng mô hình quan hệ hai ngôi. 41
4.3.1. Xác định các tính chất của quan hệ 41
4.3.2. Xác minh quan hệ tương đương. 41
4.3.3. Xác minh quan hệ tương đương. 42
CHƯƠNG 5 HỆ THỐNG GIẢI TOÁN RỜI RẠC 43
5.1. Phân tích nghiệp vụ hệ thống 43
5.2. Cấu trúc tổng quát của hệ thống 43
5.3. Giao diện chương trình 45
5.4. Công nghệ sử dụng 48
5.4.1. Nodejs 48
5.4.2. Typescript 49
5.5. Kiểm thử 49
CHƯƠNG 6 TỔNG KẾT 51

6.1. Kết quả đạt được 51

6.2. Vấn đề tồn tại 51

6.3. Kiến nghị 51

PHỤ LỤC 52

DANH MỤC CHỮ VIẾT TẮT

STT	Mô tả	Ý nghĩa	Ghi chú
1	COKB	Computational Objects Knowledge Base	
2	Expr	Expression	
3	GT	Giả thiết	
4	O	Kết luận	
5	C	Thành phần khái niệm	
6	H	Thành phần quan hệ kế thừa	

BẢNG BIỂU, SƠ ĐỒ, HÌNH VẼ

Bảng 2-1 các toán tử trong logic mệnh đề 4
Hình 2-1 Luật tương đương logic 5
Hình 2-2 Luật suy diễn 5
Bảng 3-1 Phân loại thành phần C của hàm Boolean 15
Bảng 3-2 Bìa Karnaugh 3 biến 16
Bảng 3-3 Bìa Karbaugh 4 biến 17
Bảng 3-4 Bìa Karnaugh 5 biến 17
Bảng 3-5 Quan hệ =,>=,<= 20
Bảng 3-6 Quan hệ <,> 21
Bảng 3-7 Quan hệ chia hết 21
Bảng 3-8 Quan hệ só chẵn 22
Bảng 3-9 Quan hệ ước số 22
Bảng 3-10 Quan hẹ bội số 23
Bảng 3-11 Phân loại thành phần C quan hệ 2 ngôi 24
Hình 3-1 Quan hệ kế thừa giữa các khái niệm trong quan hệ 2 ngôi 24
Bảng 4-1 Checker cải tiến của luât tương đương 29
Bảng 4-2 Checker cải tiến cho luật suy diễn 30
Bảng 4-3 Độ phức tạp toán tử 31
Bảng 4-4 Giả thiết sau khi được sắp xếp 35
Hình 4-1 Nhóm tế bào 4 của bìa Karnaugh 3 biến 38
Hình 4-2 Nhóm tế bào 2 của bìa Karnaugh 3 biến 39
Hình 4-3 Nhóm tế bào 8 của bìa Karnaugh 4 biến 39
Hình 4-4 Nhóm tế bào 4 bìa Karnaugh 4 biến 39
Hình 4-5 Nhóm tế bào 3 của biafKarnaugh 4 biến 40
Sơ đồ 5-1 Sơ đồ nghiệp vụ của hệ thống 43
Sơ đồ 5-2 Cấu trúc website giải toán rời rạc 44
Sơ đồ 5-3 Cấu trúc hệ thống giải toán 44
Sơ đồ 5-4 Cấu trúc bộ giải bài tập logic mệnh đề 44
Sơ đồ 5-5 Cấu trúc bộ giải bài tập hàm Boolean 45
Sơ đồ 5-6 Cấu trúc bộ giải bài tập quan hệ hai ngôi. 45

Hình 5-1 Màng hình nhập liệu 45
Hình 5-2 Màng hinh suy diễn logic 46
Hình 5-3 Màng hình xác định chân trị 46
Hình 5-4 Màng hình chứng minnh hai mệnh đề tương đương 47
Hình 5-5 Màng hình bài toán rút gọn hàm Boolean 47
Hình 5-6 Màng hình bài toán quan hệ hai ngôi 48
Bảng 5-1 Logo nodejs 48
Hình 5-7Logo typescript 49
Bảng 5-2 Thông kê kết quả 50

CHƯƠNG 1. MỞ ĐẦU

1.1 Giới thiệu

Ngày nay, nhu cầu về việc đưa các tri thức của con người lên máy tính, nhằm hỗ trợ cho các ứng dụng trợ giảng hay hệ thống giáo dục thông minh, đang ngày càng được quan tâm. Các tri thức khi này sẽ được tổ chức thành các Hệ cơ sở tri thức trên máy tính.

Khi tiến hành xây dựng một Hệ cơ sở tri thức thì quá trình biểu diễn tri thức đóng một vai trò đặc biệt quan trọng. Một phương pháp biểu diễn tri thức phù hợp với miền tri thức cần biểu diễn sẽ giúp việc định nghĩa hệ cơ sở trở nên dễ dàng, đầy đủ. Đồng thời, điều đó cũng giúp việc xây dựng động cơ suy diễn trơ nên hiệu quả hơn.

Ngày nay, có nhiều phương pháp biểu diễn tri thức. Chẳng hạn biểu diễn bằng mạng ngữ nghĩa, fream-based,… Điểm chung của các phương pháp này là dễ triển khai, thích hợp với một hoặc một số miền tri thức. Tuy nhiên nhược điểm của chúng là ít linh động, không có tính bao quát cao. Mặc khác cũng có một số mô hình được xây dựng chặc chẽ, nhưng yêu cầu phải dựa trên một ngôn ngữ logic nào đó, khác với các ngôn ngữ lập trình phổ biến hiện nay. Do đó, gây khó khăn cho việc cài đặt và triển khai.

Mô hình tri thức các đối tượng tính toán (Computational Objects Knowledge Base - COKB) là một ontology dựa trên phương pháp hướng đối tượng. Trong mô hình này, các miền tri thức được đặc tả bởi một đối tượng từ một class được định nghĩa sẵn. Các đối tượng này có các thuộc tính và hành vi cụ thể. Tùy thuộc vào các miền tri thức phức tạp hay không mà chúng ta sẽ có các khía cạnh khác như quan hệ giữa các tri thức, toán tử, các hàm có trong miền tri thức. Cũng chính nhờ các tiếp cận này mà mô hình COKB có thể được cài đặt trên các ngôn ngữ lập trình có hỗ trợ hướng đối tượng. Từ đó, cũng giúp việc giao tiếp giữa người thiết kế hệ cơ sở tri thức và người triển khai trở nên dễ dàng. Việc này rất thích hợp cho các dự án thực tế. Tuy nhiên, mô hình COKB vẫn còn hạn chế. Đó là việc kết hợp giữa các miền tri thức với nhau còn khó khăn. Đòi hỏi phải có một thuật toán hỗ trợ khi triển khai chúng.

Trong bày nghiên cứu này sẽ tiến hành nghiên cứu mô hình COKB và áp dụng nó vào việc xây dựng một mô hình biểu diễn tri thức mô toán rời rạc. Từ mô hình này, chúng ta sẽ tạo ra một hệ thống giúp hỗ trợ giải các bài tập trong miền tri thức này. Lời giải sẽ được trình bày một cách dễ hiểu, theo từng bước và bám sát vào các giáo trình hiện nay.

1.2 Mục tiêu nghiên cứu

Mục tiêu chính của bài nghiên cứu này, đó là tiến hành trình bày về cấu trúc mô hình ontology COKB. Áp dụng nó vào việc biểu diễn các kiến thức về miền tri thức toán rời rạc. Tiến hành xây dựng nên một website hỗ trợ việc giải các bài toán thuộc miền tri thức toán rời rạc, một cách tự động, theo từng bước và dễ hiểu. Từ đó hi vọng sẽ giúp được các bạn sinh viên ngành công nghệ thông tin nói riêng và những ai đang học môn toán rời rạc nói chung, có được một nguồn tư liệu tham khảo và học tập.

1.3 Phạm vi

Miền tri thức được trình bày trong bài nghiên cứu lần này là miền tri thức toán rời rạc. Cụ thể là giới hạn tại ba chương chính đó là : Logic mệnh đề, Hàm Boolean và Quan hệ hai ngôi.

Hệ thống giải toán rời rạc tự động được trình bày ở các phần phía dưới là một website có giao diện đơn giản dễ tiếp cận với người dùng. Trong tương lai ta có thể phát triển nó lên các nền tảng khác như di động, window,…

1.4 Cấu trúc báo cáo thực tập tốt nghiệp

1.1.1 Chương 1: Mở đầu

1.1.2 Chương 2: Tổng quan

1.1.3 Chương 3: Xây dựng mô hình cho miền tri thức toán rời rạc

1.1.4 Chương 4: Các bài toán trên miền tri thức toán rời rạc

1.1.5 Chương 5: Chương trình giải toán rời rạc

CHƯƠNG 2. TỔNG QUAN

2.1 Các kiến thức miền tri thức toán rời rạc

Toán học rời rạc (tiếng Anh: *discrete mathematics*) là tên chung của nhiều ngành toán học có đối tượng nghiên cứu là các tập hợp rời rạc. Trong bài viết này chúng ta sẽ tập trung nghiên cứu ba chương chính bao gồm: Logic mệnh đề, Hàm số Boolean và Quan hệ hai ngôi.

2.1.1 Lý thuyết Logic mệnh đề

Logic mệnh đề là một nhánh của ngành toán logic. Trong đó đối tượng nghiên cứu chính là các mệnh đề.

Mệnh đề là một câu phát biểu có chân trị đúng hoặc sai. Một mệnh đề có chân trị là 1 được xác định là một mệnh đề đúng, ngược lại mệnh đề có chân trị là 0 được gọi là mệnh đề sai. Không có mệnh đề nào có chân trị vừa đúng hoặc sai.

VD: "Hà Nội là thủ đô của Việt Nam"- là một mệnh đề với chân trị là đúng.

"Hôm nay trời lạnh quá!!" - không phải là một mệnh đề vì mang nhiều cảm tính, tùy thuộc vào mỗi người, do đó ta không thể xác định tính chất đúng hay sai cua câu nói này.

Ký hiệu: người ta thường dùng các chữ cái a,b,c,… làm ký hiệu cho mệnh đề. Chẳng hạn a= "Hà Nội là thủ đô của Việt Nam". Hoặc có thể ghi tắc là *a.*

Chú ý:

1. Các mệnh đề có chân trị đúng sai tùy thuộc vào khoảng thời gian cụ thể. Có thể trong khoảng thời gian này, mệnh đề có chân trị là đúng nhưng trong khoảng thời gian khác nó mang chân trị là sai.

   VD: "Đội tuyển Việt Nam vừa giành chiến thắng trước đối thủ của mình". Mệnh đề này sẽ có chân trị đúng hoặc sai dựa vào kết quả từng trận đấu của đội tuyển Việt Nam.

2. Có một số mệnh đề chưa xác định được chân trị tai hiện tại. Nhưng chắc chắn nó có giá trị đúng hoặc sai. VD: "Đội tuyển Việt Nam vượt qua vòng loại thứ 3 và tham dự WC 2022."

***Các toán tử trong logic mệnh đề:** Cũng tương tự các lĩnh vực toán học khác, Logic mệnh đề cũng có toán tử tác động vào các mệnh đề. Các toán tử bao gồm: nhóm toán tử một ngôi và nhóm toán tử hai ngôi. Nhóm toán tử một ngôi chỉ gồm toán tử phủ định (¬).

Nhóm toán tử hai ngôi bào gồm: $\wedge, \vee, \rightarrow, \leftrightarrow$. Kết quả của các phép toán được thể hiện đầy đủ tại bảng dưới đây.

p	q	$\neg p$	p $\vee$ q	p $\wedge$ q	p $\rightarrow$ q	p $\leftrightarrow$ q
1	1	0	1	1	1	1
1	0	0	1	0	0	0
0	1	1	1	0	1	0
0	0	1	0	0	1	1

Bảng 2-1 các toán tử trong logic mệnh đề

***Phân loại:** Dựa vào cấu trúc của mệnh đề, cho một mệnh đề p cho trước ta có nhận xét sau:

- Nếu p là mệnh đề có chân trị luôn luôn là True (đúng) hoặc False (sai) thì p được gọi là hằng.
- Nếu p là một mệnh đề đơn. p được gọi là các mệnh đề nguyên thủy hay biến mệnh đề.
- p là kết quả của các phép toán logic với các biến mệnh đề khác thì p được gọi là mệnh đề phức tạp hay biểu thức mệnh đề.

***Sự tương đương mệnh đề:**

Định nghĩa: Cho P và Q là hai mệnh đề. Ta nói rằng hai mệnh đề P, Q *tương đương logic* với nhau, nếu với mọi hệ chân lý gán cho các biến mệnh đề có mặt trong hai công thức đó thì chúng luôn nhận giá trị chân lý như nhau.

Ký hiệu: $P \equiv Q$.

Biểu thức $P \equiv Q$ được xác định thông quan một loạt các biểu thức tương đương khác. Các biểu thức này đã được chứng minh từ trước đó.

***Các luật trong logic mệnh đề** : Trong logic mệnh đề tồn tai các luật bao gồm các luật tương đương logic và luật suy diễn logic.

Mệnh đề tương đương		Tên gọi
$p \wedge True \equiv p$	$p \vee False \equiv p$	Luật đồng nhất
$p \vee True \equiv True$	$p \wedge False \equiv False$	Luật nuốt
$p \vee p \equiv p$	$p \wedge p \equiv p$	Luật lũy đẳng
$\neg(\neg p) \equiv p$		Luật phủ định kép
$p \vee q \equiv q \vee p$	$p \wedge q \equiv p \wedge q$	Luật giao hoán
$(p \vee q) \vee r \equiv p \vee (q \vee r)$	$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$	Luật kết hợp
$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$ $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$		Luật phân phối
$\neg(p \vee q) \equiv \neg p \wedge \neg q$	$\neg(p \wedge q) \equiv \neg p \vee \neg q$	Luật De Morgan
$p \wedge \neg p \equiv False$	$p \vee \neg p \equiv True$	Luật phần tử bù
$p \vee (p \wedge q) \equiv p$	$p \wedge (p \vee q) \equiv p$	Luật hấp thụ
$p \rightarrow q \equiv \neg p \vee q$		Luật phép kéo theo
$p \leftrightarrow q \equiv (p \rightarrow q) \vee (q \rightarrow p)$		Luật phép tương đương

Hình 2-1 Luật tương đương logic

Name	Rule	
Disjuntive	$p \vee q, \neg p \Rightarrow q$	
Conjunction	$p, q \Rightarrow p \wedge q$	$p \wedge q \Rightarrow p, q$
Modus Ponens	$(p \rightarrow q), p \Rightarrow q$	
Modus Tollens	$(p \rightarrow q), \neg q \Rightarrow \neg p$	
Hypothetical	$(p \rightarrow q), (q \rightarrow r) \Rightarrow (p \rightarrow r)$	
Resolution	$p \vee q, \neg p \vee r \Rightarrow q \vee r$	

Hình 2-2 Luật suy diễn

2.1.2 Lý thuyết về Hàm Boolean

Tương tự Logic mệnh đề, Hàm Boolean xoay quanh các biến mang hai giá trị 0 và 1. Tuy nhiên, điểm khác biệt giữa hai miền tri thức này đó là Logic mệnh đề dùng để biểu diễn các lý luận, suy diễn, trong khi Hàm Boolean thì không. Do đó, tại miền tri thức Hàm Boolean chỉ chứa các phép toán "+", "." và "¬" mà không chứa các dấu "→" hay

"↔". Vì vậy các quy tắc rút gọn của Hàm Boolean không có hai luật về phép tương đương và phép kéo theo. Cũng như, không có các quy tắc suy diễn logic như Logic mệnh đề.

Như vậy, hàm Boolean tập trung vào việc biểu diễn chân trị của một hàm dựa vào các trường hợp các biến mệnh đề. Do đó nó được ứng dụng nhiều trong ngành điện-điện tử, biểu diễn các mạnh điện,…

2.1.3 Quan hệ hai ngôi.

2.1.3.1 Tập hợp

Định nghĩa: Trong toán học, tập hợp là sự tụ tập của một dãy số hữu hạn hay vô hạn các đối tượng nào đó. Những đối tượng này ta gọi là phần tử của tập hợp. Một tập hợp có thể có nhiều phần tử hoặc không có phần tử nào. Ký hiệu bằng một ký tự viết hoa.

***Biểu diễn một tập hợp:** để biểu diễn một tập hợp ta có nhiều cách để thực hiện nhưng ta sẽ quan tâm đến các cách sau:

1. Phương pháp liệt kê. Nếu số lượng tập hợp là hữu hạn.

   VD: A = {1;2;3;4;5;6;7;8}.

2. Phương pháp nêu đặc tả. Phương pháp này thường dùng để đánh dấu một tập hợp mà ta không thể xác định được số lượng phần tử hoặc số lượng phần tử quá lớn, như các tập số nguyên, số tự nhiên, số thực,… đồng thời các phần tử có chung một hoặc một số đặc điểm nào đó .

   Phương pháp được hiểu như sau, cho một tập hợp không gian K cho trước. Biểu diễn tập hợp P ∈ K, sao cho các phần tử P_i của P thoản mãn mọi điều kiện của tập *Conditions.* Ví dụ:

   Số chẵn: P = SO_CHAN = {x ∈ N | x ⋮ 2 }

3. Tập hợp không có phần tử nào được gọi là tập rỗng. Ký hiệu: {} hay ∅.

2.1.3.2 Quan hệ

Định nghĩa: Một quan hệ hai ngôi từ tập A đến tập B là tập con của tích đề các R=A x B. Chúng ta sẽ viết a R b thay cho (a, b) ∈ R.

Từ định nghĩa 2.1.2 ta có các nhận xét sau về một quan hệ R:

- Một quan hệ R là một tập hợp. Do đó nó hoàn toàn có thể được biểu diễn như một tập hợp bình thường.
- Nếu quan hệ R từ chính tập A, thì quan hệ trên được gọi là quan hệ trên một tập A.

- Với các tập hợp A hữu hạn $\{a_0, a_1, \ldots, a_N\}$, ta có thể ánh xạ các phần tử (a_i, a_j) từ $a_i, a_j \in A$ lên một ma trận M NxN. Khi đó M[ij] = 1 khi $(a_i, a_j) \in R$ và ngược lại M[ij] = 0.

2.1.3.3 Các tính chất của một quan hệ: Một quan hệ sẽ có các tính cơ bản như sau.

1. **Tính phản xạ:** Xét quan hệ R và tập hợp A. Quan hệ R có tính chất *phản xạ* nếu:

$$\forall \, a \in A \, , \, a \, R \, a$$

2. **Tính đối xứng:** Xét quan hệ R và tập hợp A. Quan hệ R có tính chất *đối xứng* nếu:

$$\forall \, a,b \in A, \;\; (a \, R \, b) \rightarrow (b \, R \, a)$$

3. **Tính phản xứng:** Xét quan hệ R và tập hợp A. Quan hệ R có tính chất *phản xứng*:

$$\forall \, a,b \in A, \;\; (a \, R \, b) \wedge (b \, R \, a) \rightarrow (a = b)$$

4. **Tính bắc cầu:** Xét quan hệ R và tập hợp A. Quan hệ R có tính chất *bắc cầu* nếu:

$$\forall \, a,b,c \in A, \;\; (a \, R \, b) \wedge (b \, R \, c) \rightarrow (a \, R \, c)$$

2.1.3.4 Phân loại quan hệ hai ngôi:

Một quan hệ hai ngôi có thể được phân ra làm hai loại bao gồm: quan hệ tương đương và quan hệ thứ tự.

**** Quan hệ tương đương:***

Định nghĩa: Quan hệ tương đương: Cho quan hệ R trên A, R được gọi là tương đương nếu R có các tính chất: phản xạ, đối xứng và bắc cầu.

VD: Quan hệ R trên tập số nguyên sao cho với hai số nguyên a,b bất kỳ ta luôn có a+b là số chẵn. Là một quan hệ tương đương vì chứa đủ ba tính chat phản xạ, đối xứng và bắc cầu.

Lớp tương đương: Cho R là quan hệ tương đương trên A và phần tử $a \in A$. Lớp tương ương chứa a được ký hiệu bởi $[a]_R$ hoặc [a] là tập:

$$[a]_R = \{b \in A | \; b \, R \, a\}$$

Dựa theo định nghĩa lớp tương đương ta có được định lý như sau, cho R là quan hệ tương đương trên tập A và a,b ∈ A, Khi đó:

1. a R b nếu $[a]_R = [b]_R$.
2. $[a]_R \neq [b]_R$ nếu $[a]_R \cap [b]_R = \emptyset$.

Như vậy ta nhận thấy tập A có thể phân hoạch thành các lớp tương đương.

**** Quan hệ thứ tự:***

Định nghĩa : Quan hệ R trên tập A là quan hệ thứ tự (thứ tự) nếu nó có tính chất phản xạ, phản xứng và bắc cầu. Ký hiệu "≺".

Tính chất:

1. Với x,y ∈ A và x ≺ y thì khi đó ta nói y là trội của x (x bị trội bởi y)
2. Trường hợp y là trội trực tiếp của x nếu y là trội của x và không tồn tại một trội z nào sao cho z bị trội bơi y và z là trội của z.
3. Xét phần tử x ∈ A :

- x gọi là phần tử tối đại, nếu x không bị trội bởi bất kỳ phần tử nào khác trong A.
- x được gọi là phần tử tối tiểu nếu x là trội của bất kỳ phàn tử trong A.
- x được gọi là giá trị lớn nhất nếu x là trội của mọi phần tử trong A
- x được gọi là giá trị nhỏ nhất nếu x bị trội bởi mọi phần tử trong A

2.2 Mô hình COKB

Như đã đề cập ở phần trước đó. Mô hình COKB được xây dựng dựa trên phương pháp hướng đối tượng. Do đó, các thành phần của mô hình sẽ được biểu diễn bởi các đối tượng. Các thành phần này bao gồm:

K = (C, H, R, Ops, Funcs, Rules)

Trong đó:

+ C là một tập hợp các khái niệm về các C-Object
+ H là một tập hợp các quan hệ phân cấp giữa các loại đối tượng.
+ R là tập hợp các khái niệm về các loại quan hệ trên các C-Object.
+ Ops là một tập hợp các toán tử.

+ Funcs là một tập hợp các hàm.

+ Rules là tập hợp các luật.

Các thành phần này, tùy thuộc vào các miền tri thức được biểu diễn, mà sẽ được xuất hiện hay không. Với miền tri thức toán rời rạc (lưu ý: chỉ bao gồm 3 chương là logic mệnh đề, hàm Boolean và quan hệ hai ngôi). Miền tri thức K của chúng ta sẽ là

$$K = K_L + K_B + K_R$$

Với K_L, K_B, K_R lần lượt là các miền tri thức của Logic mệnh đề , Đại số Boolean và Quan hệ hai ngôi.

CHƯƠNG 3. XÂY DỰNG MÔ HÌNH CHO MIỀN TRI THỨC TOÁN RỜI RẠC

3.1 Xây dựng mô hình cho miền tri thức Logic mệnh đề

Đối với miền tri thức của Logic mênh đề, chúng ta sẽ xây dựng cấu trúc mô hình như sau:

$$K = (C,Ops,Rules) + Funcs$$

3.1.1 Cấu trúc (C,Ops,Rules)

C là tập hợp các khái niệm trong miển tri thức được biểu diễn. Ở đây, miền tri thức Logic mệnh đề chỉ có một khái niệm duy nhất đó là biểu thức mệnh đề.

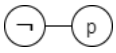
Định nghĩa về một biểu thức mệnh đề:

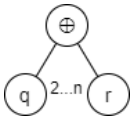
1. Các giá trị 1 (True) hoặc 0 (False) là một biểu thức mệnh đề, và chúng được gọi là một biểu thức hằng
2. Một biến p, chỉ có hai giá trị là 0 hoặc 1 là một là một biểu thức mệnh đề. Nó được gọi là một biến mệnh đề.
3. Nếu p là một biểu thức mệnh đề thì khi đó $\neg p$ cũng là một biểu thức
4. Với p và q là các biểu thức mệnh đề thì p $\oplus$ q là một biểu thức mệnh đề (với $\oplus$ là các phép toán $\wedge, \vee, \rightarrow, \leftrightarrow$).

Biểu diễn cấu trúc biểu thức mệnh đề:

Cho một biểu thức p. Khi đó cây T(p) biểu diễn p như sau:

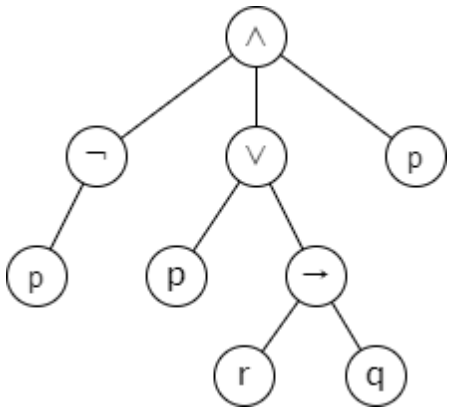
1. Nếu p là một hằng hoặc p là một biến mệnh đề thì T(p) = 

2. Nếu p = ¬q thì T(p) = 

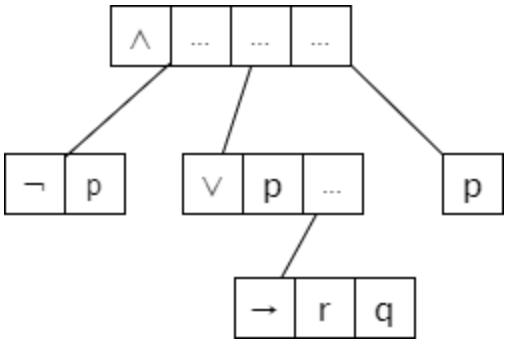
3. Nếu p = q $\oplus$ … $\oplus$ r thì T(p) = 

4. Nếu p là biểu thức con của q thì T(p) là nhánh con của T(q)

Ví dụ biểu thức $A = \neg p \wedge (p \vee (r \rightarrow q)) \wedge p$ được biểu diễn như sau:



Hoặc ta có thể viết gọn thành:



Lúc này khi duyệt cây trên, ta hoàn toàn có thể biểu thức trên lại thành dạng hậu tố:

$$A = (\wedge,(\neg,p),(\vee,p,(\rightarrow,r,q)),p) = (\wedge,B,C,p).$$

Trong đó: $B=(\neg,p)$, $D=(\rightarrow,r,q)$, $C=(\vee,p,D)$

Tuy nhiên cách thức này sẽ phát sinh ra một nhược điểm nhỏ đó là việc biểu diễn sẽ gom các mệnh đề có cùng toán tử lại với nhau. Do đó, khi biểu diễn các mệnh đề có nhiều toán tử ta cần phải nhóm các mệnh đề lại bằng cặp dấu "( )", nhằm tạo ra các biểu thức con. Chẳng hạn, ta không nên đưa một mệnh đề như thế này: $P = a \vee b \wedge c$,mà hãy nhóm nó lại thành $P = a \vee (b \wedge c)$ hoặc $P = (a \vee b) \wedge c$. Điều này cũng hoàn toàn hợp với ngôn ngữ giao tiếp thường ngày của chúng ta.

Thành phần C của chúng ta sẽ được xây dựng dựa trên cấu trúc trên. Cụ thể, thành phần C biểu diễn một biểu thức mệnh đề có dạng như sau:

C = (Attr, M_Funcs, Parent)

Trong đó, *M_Funcs* là các hàm hành vi của một biểu thức mệnh đề. Bao gồm các phương thức quan trọng đó là ***getTruthValue()*** để lấy chân trị của một mệnh đề. Hàm ***getID()*** lấy chuỗi hậu tố của của biểu thức đó. ***getSubExpr()*** lấy mảng biểu thức con của C. Nếu C là biến, trả về **NULL**. *Parent* là con trỏ tham chiếu đến biểu thức chứa *C*. Nếu *C* là biểu thức gốc *Parent* là **NULL**

Thành phần Attr của sẽ được cấu tạo bao gồm

- Truth_Value: một biến Boolean, thể hiện chân trị một biến mệnh đề hay hằng
- Id: là ký tự đại diện cho biến mệnh đề, với hằng thì ký tự này là 1 hoặc 0
- O: là toán tử logic của một biểu thức
- c: là tập các biểu thức con cảu biểu thức đang xét.
- Primes: là tập các biến mệnh đề cấu tạo nên biểu thức.

Tiếp theo là thành phần *Ops*. Đây là tập hợp định nghĩa về các toán tử trong miền tri thức Logic mệnh đề.Như đã đề cập tại mục [I]. Các toán tử này sẽ đảm nhận vai trò tính toán giá trị của các biểu thức mệnh đề mà chúng tác động đến. Chúng tương đương với các toán tử mà ta đã đề cập trước đó

Thành phần cuối cùng là *Rules*. Là các quy tắc rút gọn và suy diễn trong phần trước.

Trong phần này chúng ta sẽ tiến hành đặc tả cho các luật trong Logic mệnh đề:

*** Đặc tả luật tương đương** có dạng :

$p \equiv q$, trong đó p và q là hai biểu thức mệnh đề.

Như vậy ta tiến hành xây dựng một đặc tả cho mỗi một luật như sau:

R = (Name, Variables, Left, Right, Checker)

Với *Name*: là tên của R,

Variables: tập các biến mệnh đề cấu thành nên luật,

Left: là thành phần biểu thức nằm bên trái R (ngẫn cách bởi dấu tương đương),

Right: là thành phần biểu thức nằm bên phải R (ngẫn cách bởi dấu tương đương),

Checker: là một đối tượng giúp kiểm tra R có áp dụng cho một biểu thức hay không. Các hàm kiểm tra của Checker có thể được Override, nhằm đáp ứng các nhu cầu sau này.

∗ Luật tương đương có dạng như sau:

$A[a_1,a_2,...,a_n] \rightarrow B[b_1,b_2,...,b_n]$, với A,B là các tập biểu thức

Hay $(a_1 \wedge a_2 \wedge ... \wedge a_n) \rightarrow (b_1 \wedge b_2 \wedge ... \wedge b_n)$

Xây dựng cấu trúc luật suy diễn:

R = (Name, Variables, Hypothesis, Goal, Checker)

Trong đó:

Name: là tên của R,

Variables: tập các biến mệnh đề cấu thành nên luật,

Hypothesis: tập các biểu thức giả thiết

Goal: tập các biểu thức kết quả,

Checker: là một đối tượng giúp kiểm tra R có áp dụng cho một biểu thức hay không. Các hàm kiểm tra của Checker có thể được Override, nhằm đáp ứng các nhu cầu sau này.

3.1.2 Thành phần Funcs

Định nghĩa các hàm dùng trong miền tri thức:

1. **Checker(P,R).** Đây là hàm dùng để kiểm tra biểu thức *P* có áp dụng cho luật *R* hay không. Trả về một biểu thức, nếu không áp dụng được trả về NULL. Cấu trúc cụ thể được đề cập mục sau.
2. **Replace(p,q,r):** hàm tiến hành thay thế biểu thức p bằng q trong biểu thức gốc r.
3. **Replace(A[a_1, $a_{1,...,}$ a_n],B[b_1, $b_{2,...,}$ b_n],p):** hàm thay thế một tập các biểu thức con *A* bằng một tập các biểu thức con *B* trong biểu thức gốc *p*.
4. **Copy(p) :** tạo ra một bản sao của biểu thức p.
5. **Length(p):** hàm lấy độ phức tap của một biểu thức. Nhằm so sánh trong quá trình đơn giản hóa.
6. **Contain(p,q):** kiểm tra biểu thức p có chứa biểu thức q hay không.
7. **Not(p):** phủ định một biểu thức.
8. **Simplify(p):** hàm rút gọn *p*, trả về *g* và các bước chuyển đổi *cd*

9. **Equivalent (p,q):** chứng minh hai mệnh đề tương đương, nếu có trả về tập ChuyenDoi [], nếu không trả về [].

3.2 Xây dựng mô hình cho miền tri thức đại số Boolean

Dựa vào các kiến thức lý thuyết của chương đại số hàm Boolean, ta sẽ tiến hành biểu diễn miền tri thức theo mô hình sau:

$$K = (C,Ops,Rules)+Funcs$$

Định nghĩa đại số Boolean:

1. Các hằng 0 và 1 là một Boolean
2. Các biến a,b,c,…. Cũng là một Boolean
3. Nếu p và q là các biểu thức thì p.q , p+q và ¬p cũng là Boolean.

Như vậy ta thấy rằng, đại số Boolean khá tương đồng với Logic mệnh đề.

Định nghĩa Hàm Boolean:

Hàm Boolean là một ánh xạ sao cho:

$$f:B^n \rightarrow B, \text{ trong đó } B = \{0,1\}$$

Vậy hàm Boolean là một hàm số có dạng: $f(x_1,x_2,...,x_n)$, với $x_1,x_2,...,x_n$ là các biến chỉ nhận hai giá trị 0 và 1, f cũng nhận giá trị 0 và 1.

Ta ký hiệu F_n để đại diện cho một hàm Boolean có n biến.

Lưu ý: Cũng vì Hàm Boolean là một hàm n biến, nên ta có thể dễ dàng biểu diễn chúng qua một đa thức. Có nhiều cách để biểu diễn một hàm Boolean, tuy nhiên để theo chuẩn của các giáo trình giảng dạy trong các trường đại học, bài nghiên cứu này chỉ xét việc biểu diễn hàm Boolean thông qua *dạng nối rời chính tắc* của nó.

Định nghĩa về dạng nối rời chính tắc:

Xét một hàm Boolean F_n có n biến:

1. Đối các biến x_i và phủ định của nó là ¬x được gọi là một *từ đơn*
2. *Đơn thức* là tích khác không của một số hữu hạn các *từ đơn*.
3. *Từ tối tiểu* là một đơn thức nhưng có đầy đủ n *từ đơn*
4. *Dạng nối rời chính tắc* là công thức biểu diễn F_n thành tổng của các *từ tối thiểu*.

Sau khi trình bày về cấu trúc của một hàm Boolean ta sẽ tiến hành xây dựng mô hình cho nó.

3.2.1 Thành phần C

Đối với đại số Boolean, đối tượng chính mà chúng ta làm việc tới đó chính là hàm Boolean. Cũng bởi vì cách thức biểu diễn của hàm Boolean thông qua các dạng nối rời chính tắc đã được nêu cụ thể ở phần trước. Do đó, ta nhận thấy các khái niệm cần lưu ý sẽ được nêu dưới đây (các khái niệm cáp độ càng cao sẽ bao phủ các cấp độ cấp dưới).

Cấp độ	Tên gọi
0	Từ đơn
1	Đơn thức
2	Dạng chuẩn tắc

Bảng 3-1 Phân loại thành phần C của hàm Boolean

Lưu ý:

1. Chúng ta cần chú ý rằng, Đại số Boolean tương tự với miền tri thức Logic mệnh đề. Do đó cấu trúc C của Logic mệnh đề hoàn toàn có thể bao quát được các khái niệm trong thành phần C của Đại số Boolean. Mà không cần phải phân cấp bậc.
2. Ở đây, chúng ta chỉ xét đa thức đã ở dạng chuẩn tắc. Do đó nếu đa thức cần biểu diễn không ở dạng chuẩn tắc thì cần phải xử lý trước khi tiến hành biểu diễn.
3. Đôi khi ở nhiều đề bài, các đa thức không nằm hoàn toàn ở dạng nối rời chuẩn tắc. Mà chỉ đưa ở dạng chuẩn tắc mà thôi. Tức là có thể ở đơn thức sẽ thiếu mất một số từ đơn, nhưng nhìn chung đa thức vẫn giữ ở dạng "tổng của các tích"

*** Từ đơn:**

Cấu trúc có dạng: *C = (sign,name,value,m_ Funcs).*

Trong đó: sign là chỉ dấu trong từ này, có phủ định hay không.

name là tên của từ đơn.

value là chân trị của nó.

m_ Funcs là tập các phương thức như ge, set,..

*** Đơn thức:**

Cấu trúc: C = *(c, m_Funcs)*. Trong đó, c là tập hợp các từ đơn. m_Funcs là tâp các phương thức quan trọng như lấy chân trị, lấy chuỗi hậu tố,…

*** Đa thức chuẩn tắc:**

Cấu trúc: C = *(V,M, m_Funcs)*. Trong đó, V là tập hợp các biến (từ đơn), M là các đơn thức. m_Funcs là tâp các phương thức quan trọng như lấy chân trị, lấy chuỗi hậu tố,…

3.2.2 Thành phần Ops

Thành phần toán tử Ops biểu diễn toán tử trong đại số Boolean cụ thể là "+", "-" và "¬".

3.2.3 Thành phần Rules bao gồm các bộ quy tắc sau:

** Quy tắc thứ tự Bìa Karnaugh:*

Để có thể ánh xạ một đa thức chuẩn tắc lên một bìa Karnaugh cần các quy tắc cụ thể. Bìa Karnaugh là tập hợp một hoặc nhiều ma trận với một số thứ tự cụ thể.

-Với hàm Boolean 3 biến ta có bìa Karnaugh như sau:

	$\overline{AB}$	$\overline{A}$ B	A B	A $\overline{B}$
$\overline{C}$	0	2	6	4
C	1	3	7	5

Bảng 3-2 Bìa Karnaugh 3 biến

-Với hàm Boolean 4 biến ta có bìa Karnaugh như sau:

	$\overline{AB}$	$\overline{A}$ B	A B	A $\overline{B}$
$\overline{CD}$	0	4	1	8

			2	
$\overline{C}$ D	1	5	13	9
CD	3	7	15	11
$C\overline{D}$	2	6	14	10

Bảng 3-3 Bìa Karbaugh 4 biến

-Với hàm Boolean 5 biến ta có bìa Karnaugh như sau:

	$\overline{BC}$	$\overline{B}C$	BC	$B\overline{C}$		$\overline{BC}$	$\overline{B}C$	BC	$B\overline{C}$
$\overline{DE}$	0	4	12	8		16	20	28	24
$\overline{D}E$	1	5	13	9		17	21	29	25
DE	3	7	15	11		19	23	31	27
$D\overline{E}$	2	6	14	10		18	22	30	26
$\overline{A}$						A			

Bảng 3-4 Bìa Karnaugh 5 biến

Để có thể ánh xạ các đa thức chính tắc, ta cần phải xem số lượng các từ đơn trong hàm Boolean. Sau đó xét từng đơn thức để ánh xạ chúng lên bìa.

VD: Giả sử cho một hàm Boolean 4 biến f, có đơn thức abc$\overline{d}$. Ta có thể ánh xạ nó lên bìa Kar(f):

$$abc\overline{d} \xrightarrow[nhị\ phân]{} 1110 \xrightarrow[thập\ phân]{} = 14.$$

Vậy abc$\overline{d}$ nằm ở ô có đánh số 14 ở Kar(f).

**Quy tắc tế bào lớn của bìa Karnaugh:*

Định nghĩa tế bào lớn: Cho một hàm Boolean f với n biến, Kar(f) là bìa Karnaugh ánh xạ từ hàm f, ta có các định nghĩa sau:

1. Tế bào T là một hình chữ nhật gồm 2^n ô và T ∈ Kar(f).
2. Tế bào T là một tế bào lớn nếu T thõa hai điều kiện:

- T là một tế bào và T ∈ Kar(f).
- Không tồn tại T' nào sao cho T' ≠ T và T ∈ T' ∈ Kar(f).

Từ các định nghĩa trên ta có nhận xét:

1. Một từ tối thiểu thuộc hàm f, thì khi ánh xạ sẽ là một tế bào T của bìa Kar(f).
2. Để nhóm các tế bào thành một tế bào lớn hơn, các tế bào cần có các từ đơn giống nhau.

3.2.4 Thành phần Funcs

Bào gồm các hàm chính sau:

Cell : ánh xạ một đơn thức lên một ô trong kar(f).

TeBaoLon: Sàng lọc lấy các tế bào lớn của kar(f).

Binary2Express: Chuyển đổi một đơn thức sang nhị phân.

3.3 Quan hệ hai ngôi

Trong bài nghiên cứu này, các quan hệ hai ngôi mà chúng ta xem xét được nằm gói gọn trong tập không gian số nguyên Z. Tức là cấu trúc của các tập hợp hay quan hệ đều là các số nguyên.

Cần nhắc lại rằng, các quan hệ hai ngôi là các tập hợp (vô hạn hoặc hữu hạn), chúng có hai cách để biểu diễn. Đó là liệt kê phần tử và nêu đặc tả của chúng. Và cũng vì tập không gian là tập số nguyên Z, do đó khi liệt kê phần tử chính là liệt kê ra hữu hạn các số nguyên và khi nêu ra đăc tả là việc trình bày một *quan hệ đại số* của phần tử trong tập hợp.

Dựa vào nhận định trên , ta có thể xây dựng một mô hình có cấu trúc có dạng:

$$K=(C,H,Rules)+Funcs+K_{SoNguyen}$$

Lưu ý:

- C: là tập hợp các khái niệm trong miền tri thức quan hệ hai ngôi.
- H: trình bày quan hệ mang tính kế thừa giữa các khái niệm
- Rules: thể hiện các tính chat của một quan hệ hai ngôi, bao gồm 4 tính chất.
- Funcs: tập hợp các hàm được dùng trong miền tri thức.
- $K_{SoNguyen}$ là miền tri thức về Biểu thức số nguyên và miền tri thức về quan hệ số nguyên.

3.3.1 Thành phần $K_{SoNguyen}$

$$K=(C,Rules)+ Funcs$$

**Thành phần C,* thành phần này gồm hay khái niệm chính: là biểu thức đại số và quan hệ số nguyên.

Biểu thức đại số:

Do đây chỉ là một thành phần phụ, không đào sâu vào nó.Thêm nữa, các biểu thức chỉ chứa các phép toán "+,-,*, / ".

$$C = (attr,childs,parent,m\_funcs)$$

Trong đó:

- *Childs* là các biểu thức con.
- *Parent* là biểu thức cha.
- *M_funcs* tập các phương thức có trong biểu thức.
- *Attr* bao gồm:

+ *sign*: thể hiện dấu của biểu thức. Chỉ áp dụng cho hằng và biến.

+ *operator*: thể hiện phép toán, +,-,*,/ .

+ *id*: ký tự character hoặc số đại diện cho nó. Với một biểu thức phức hợp thì id là một chuỗi hậu tố của nó.

+ *value*: giá trị được gáng vào, chỉ áp dụng cho biến và hằng.

Định nghĩa: *Biểu thức con của một biểu thức*.

Gọi p là một biểu thức, khi đó q và r là các biểu thức con của p nếu p = q $\otimes$ r ($\otimes$ là các phép +,-,*,/). Lưu ý các thứ tự và cấp bậc biểu thức dựa vào chuỗi hậu tố của biểu thức mà ta đang xét.

Quan hệ đai số

Định nghĩa: *Các quan hệ trong tập số nguyên thường có dạng như sau.*

<expr1> ⊗ <expr2> hoặc #<expr1>

Trong đó: *expr1, expr2* là các biểu thức số nguyên.

"⊗, #" là sự ràng buộc về mặt toán học giữa các biểu thức expr1, expr2.

Như vậy thành phần C có cấu tạo:

C = Left, Name, Right, Checker, M_Rules.

Với Left là một biểu thức bên trái của ràng buộc, Right là một biểu thức nằm phải của ràng buộc. Name là ký hiệu hoặc tên gọi tắt của ràng buộc đó. Checker là tập các phương pháp nhằm kiểm tra với hai biểu thức p,q bất kỳ nào gáng cho Left và Right, thì ràng buộc đang xét có hợp lệ hay không. Cuối cùng, M_Rules là bộ các tính chất mà ràng buộc đó có trong số nguyên.

Dựa vào việc thống kê các bài tập quan hệ hai ngôi. Chúng ta sẽ có các dạng ràng buộc thường gặp như sau:

- Ràng buộc ⊗ = "=", ">=", "<="

Ký tự	"=", "<=" , ">="
Biểu diễn	expr1 ⊗ expr2
Ý nghĩa	epxr1 bằng (lớn hơn hoặc bằng, bé hơn hoặc bằng) expr2
Điều kiện	- expr1.Value ⊗ expr2.Value - expr1.Id = expr2.Id
Tính chất	- Với a , thì a ⊗ a. - Nếu: a ⊗ b, thì b ⊗ a - Nếu: a ⊗ b và b ⊗ c thì a ⊗ c

Bảng 3-5 Quan hệ =,>=,<=

- Ràng buộc ⊗ = ">", "<"

Ký tự	>,<
Biểu diễn	expr1 ⊗ expr2
Ý nghĩa	epxr1 lớn hơn(nhỏ hơn) expr2
Điều kiện	- expr1.Value ⊗ expr2.Value
Tính chất	- Nếu: a ⊗ b và b ⊗ c thì a ⊗ c

Bảng 3-6 Quan hệ <,>

- Ràng buộc ⊗ = "Chia hết"

Ký tự	":"
Biểu diễn	expr1 ⊗ expr2
Ý nghĩa	epxr1 chia hết expr2
Điều kiện	- expr1.Value mod expr2.Value = 0 - expr1.Value = 0 - expr1 = k * expr2, với k là một biểu thức tùy ý
Tính chất	- Với a , thì a ⊗ a và -a ⊗ a. - Nếu a ⊗ b và b ⊗ c thì a ⊗ c. - Nếu a ⊗ c và b ⊗ c thì a+b ⊗ c, a-b ⊗ c , a*b ⊗ c .

Bảng 3-7 Quan hệ chia hết

- Ràng buộc # = "Là số chẵn"

Viết tắt	is Odd
Biểu diễn	# expr1

Ý nghĩa	epxr1 là một số chẵn
Điều kiện	- expr1.Value mod 2 = 0 - expr1 = 2*k, với k là một biểu thức tùy ý
Tính chất	- Nếu #a thì # (-a). - Nếu #a và #b thì #(a+b), #(a-b), #(a*b).

Bảng 3-8 Quan hệ só chẵn

- Ràng buộc ⊗ = "là ước số của"

Viết tắt	Is the divisor of
Biểu diễn	expr1 ⊗ expr2
Ý nghĩa	epxr1 là ước của expr2
Điều kiện	- expr2.Value mod expr1.Value = 0 - expr2.Value = 0 - expr2 = k * expr1, với k là một biểu thức tùy ý
Tính chất	- Với a , thì a ⊗ a và -a ⊗ a. - Nếu a ⊗ b và b ⊗ c thì a ⊗ c. - Nếu b ⊗ a và c ⊗ b thì c ⊗ a - Nếu a ⊗ c và b ⊗ c thì a+b ⊗ c, a-b ⊗ c , a*b ⊗ c .

Bảng 3-9 Quan hệ ước số

- Ràng buộc ⊗ = "là bội số của"

Viết tắt	Is the multiple of
Biểu diễn	expr1 ⊗ expr2
Ý nghĩa	epxr1 là bội của expr2

Điều kiện	- expr1.Value mod expr2.Value $= 0$ - expr1.Value $= 0$ - expr1 $=$ k * expr2, với k là một biểu thức tùy ý
Tính chất	- Với a , thì a $\otimes$ a và -a $\otimes$ a. - Nếu a $\otimes$ b và b $\otimes$ c thì a $\otimes$ c. - Nếu b $\otimes$ a và b $\otimes$ c thì a $\otimes$ c. - Nếu a $\otimes$ c và b $\otimes$ c thì a+b $\otimes$ c, a-b $\otimes$ c , a*b $\otimes$ c .

Bảng 3-10 Quan hệ bội số

3.3.2 Thành phần (C,H,Rules)

**Thành phần C*:

C = Attr, m_Funcs, m_Rules

Trong đó: Attr là tập các thuộc tính của khái niêm, m_Rules là các quy tắc trong nội tại khái niệm.

C bao gồm các khái niệm quan trọng, các khái niệm sau được kế thừa được các đặc điểm khái niệm trước đó:

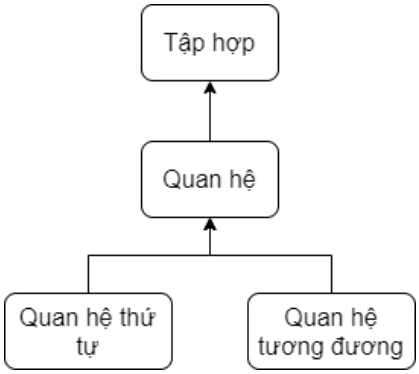
Cấp	**Tên khái niệm**	**Attr**	**M_Rules**
0	Tập hợp	-*elements:* tập các phần tử. -*condition:* đặc tả bằng "quan hệ đại số" -*id:* chuỗi biểu diễn tập hợp: + Nếu tập hợp có hữu hạn các element thì chuỗi này bao gồm các phần tử element + Nếu tập hợp là dạng đặc tả, thì id là một cấu trúc đặc tả.	Không có
1	Quan hệ	-*Không gian mẫu:* là một tập hợp, nó đại diện cho tập hợp mà quan hệ	Không có

		đang xét tới. *-Tính chất:* là các tính chất của một quan hệ hai ngôi. Gồm 4 tính chất.	
2	Quan hệ tương đương	*-Lớp tương đương:* Tập hợp các lớp tương đương của quan hệ tương đương	Quy tắc về phân hoạch tập hợp.
2	Quan hệ thứ tự	*-Hasse* *- Giá trị lớn nhất, nhỏ nhất* *- Tối tiểu , tối đại.*	Quy tắc về các phần tử là tối tiểu và tối đại

Bảng 3-11 Phân loại thành phần C quan hệ 2 ngôi

**Thành phần H*: miêu tả quan hệ kế thùa của các khái niệm được diễn ta bởi bảng sau đây.

Hình 3-1 Quan hệ kế thừa giữa các khái niệm trong quan hệ 2 ngôi



**Thành phần Rules:* là các tính chất của quan hệ 2 ngôi. Do một quan hệ có thể được biểu diễn bởi hai cách nên cũng có hai tập luật dành cho bốn tính chất này. Bao gồm bộ luật dành cho quan hệ cấu trúc dạng danh sách và bộ luật dành cho quan hệ quy định bằng đặc tả.

Bộ luật cho tập hợp có cấu trúc danh sách: Gọi M là ma trận được ánh xạ tử quan hệ R và R có tập mẫu là A. Khi đó các tính chất của R được xác định bằng.

- Tính phản xạ: $M_{ii} = 1$, vơi mọi $i \in A$.
- Tính đối xứng: Nếu $M_{ij} = 1$ thì $M_{ji} = 1$, vơi mọi $i,j \in A$.

- Tính phản xứng: Tính phản xạ phải hợp lệ và không tồn tại i,j ∈ A sao cho $M_{ij} = 1$ thì $M_{ji} = 1$
- Tính bắc cầu: Nếu $M_{xy} = 1$ và M_{yz} thì $M_{xz} = 1$, vơi mọi x,y,z ∈ A.

Bộ luật cho tập hợp quy định bằng đặc tả:

- Tính phản xạ: Nếu trong R thay thế b bằng a thì *R.condition* phải đúng.
- Tính đối xứng: Giả sử R là đúng, R2 là quan hệ hai ngôi được tạo ra bằng cách *Replace(a,b,R.condition)* và *Replace(b,a,R.Condition).* Thì khi đó:

+ Hoặc R2.Condition là đúng.

+ Hoặc SuyDien(Giả thiết = R, Kết luận = R2) là đúng.

- Tính phản xạ: tính phản xạ là hợp lệ đồng thời tính đối xứng là sai.
- Tính bắc cầu: Giả sử

$$R1 = \begin{cases} Replace(a, a, R.condition) \\ Replace(b, b, R.condition) \end{cases}$$

$$R2 = \begin{cases} Replace(a, b, R.condition) \\ Replace(b, c, R.condition) \end{cases}$$

$$R3 = \begin{cases} Replace(a, b, R.condition) \\ Replace(b, c, R.condition) \end{cases}$$

R1,R2 là đúng thì hoặc *R3.condtion* là đúng hoặc SuyDien(Giả thiết = {R1,R2}, Kết luận = R3).

3.3.3 Thành phần Funcs.

Phương thức quan trọng gồm:

- Postfix: sinh ra chuỗi hậu tố từ một chuỗi biểu thức.
- String2Expession*:* Chuyển từ một chuỗi hậu tố sang một đối tượng biểu thức.
- Symplify*:* Rút gọn một biểu thức.
- SuyDien: Suy diễn một quan hệ 2 ngôi.

CHƯƠNG 4. XÂY DỰNG THUẬT GIẢI CHO CÁC VẤN ĐỀ TRÊN MIỀN TRI THỨC TOÁN RỜI RẠC

Trong các chương trình giảng dạy môn toán rời rạc, ở các chương logic mệnh đề, đại số Boolean và quan hệ hai ngôi, ta sẽ có các dạng bài tập chính sau:

- Rút gọn biểu thức logic
- Suy diễn logic
- Xác định chân trị biểu thức logic
- Tối giản hóa hàm Boolean
- Xác định các tính chất của một quan hệ hai ngôi.
- Xác định một quan hệ hai ngôi có là quan hệ tương đương hay không và đưa ra các lớp tương đương
- Xác định một quan hệ có là quan hệ thứ tự hay không và đưa ra các thuộc tính của nó.

4.1 Mô hình cho cái bài toán logic mệnh đề

Trong miền tri thức về logic mệnh đề, hai vấn đề chính xoay quanh đó là sự tương đương logic và sự suy diễn logic.

4.1.1 Sự tương đương logic

Định nghĩa: Cho P và Q là hai biểu thức mệnh đề. Ta nói rằng hai công thức P, Q tương đương logic với nhau, ký hiệu là P ≡ Q, nếu với mọi hệ chân lý gán cho các mệnh đề có mặt trong hai công thức đó thì chúng luôn nhận giá trị chân lý như nhau.

Tính chất:

1. Nếu biểu thức p ≡ q thì q ≡ p.
2. Cho một luật r có các biến variables (được nêu ở chương trước đó). Nếu ta thay thế các biến trong r bằng các biểu thức khác. Thì ta nhận được một biểu thức mới, tương đương r.
3. Cho một biểu thức *f* ,có chứa một biểu thức con *g*. Nếu ta thay thế *g* bằng một biểu thức, thì cũng tạo ra một biểu thức mới tương đương với *f*
4. Cho biểu thức *f* và *g* .Nếu qua k lần biến đổi *f* theo các luật tương đương ta nhận một biểu thức mới chính là *g*, thì ta nói *f* và *g* tương đương nhau

Định nghĩa: một biểu thức *p* được gọi là *một trạng thái* của *q* nếu *p* tương đương *q*. Như vậy một biểu thức *p* tương đương với *q*, có thể được hiểu như sau:

$$p \equiv \underbrace{p_1 \equiv p_2 \equiv \cdots \equiv p_n}_{n} \equiv q$$

hoặc: $p \overset{P_T}{\rightarrow} q$

Trong đó: p_1, $p_{1,...,}$ p_n là các trạng thái trung gian, sinh ra bằng cách chuyển đổi *p* qua các luật tương đương. P_T là các trạng thái của *p* và *q*.

Định nghĩa: *Bộ kiểm tra Checker cho luật tương đương*

Checker luật tương đương, ký hiệu $Cheker_{ER}$, là một thành phần của một đối tượng luật tương đương. Nó giúp kiểm tra một luật tương đương có thể áp dụng vào một biểu thức nào không, từ đó biến đổi nó thành một *trạng thái* mới.

Ý tưởng triển khai một checker cho một luật *r* như sau. Ta xây dựng *r* như một biểu thức bình thường. Xét một biểu thức *p,* khi này ta cần thay thế dần các biến mệnh đề trong *p* vào *r* hoặc thay các biến trong *r* lần lượt bằng các biểu thức con trong *p*. Nếu sau khi thay thế có thể biến *r* thành một biểu thức mới giống với *p* thì ta nói *r* có thể áp dụng vào *p*.

VD: P = x ∨ x , P có thể áp dụng luật lũy đẳng (p ∨ p).

P = (x ∨ y) ∨ (x ∨ y), P có thể áp dụng lũy đẳng (p ∨ p).

Tuy nhiên đôi khi một số luật không nhất thiết phải thay thê và kiểm tra như vậy. Mà chúng ta chỉ cần xem xét các *đặc điểm* của chúng, chẳng hạn với luật kéo theo: a → b ≡ ¬x∨y. Nếu muốn kiểm tra biểu thức với vế trái của luật này, ta chỉ cần xem biểu thức có chứa dấu kéo theo hay không. Nhờ vậy, ta có thể tối ưu cho quá trình kiểm tra.

Do đó, bộ kiểm tra checker là sự kết hợp giữa các hàm kiểm tra thông thường và các hàm kiểm tra cải tiến như sau:

$$Checker_{ER} = Checker_{Default} \cup Checker_{Advance}$$

Định nghĩa: *Bộ checker mặc định.*

1. Cho một biểu thức *p* và luật tương đương *R*. Nếu:

+ *R.Left.getID() == p.getID()* thì đặt *p = q = R.Right*

+ *R.Right.getID() == p.getID()* thì đặt *p = q = R.Left*

2. Cho một biểu thức *p*, *p* có một mảng các biểu thức con *C*. Luật *R*, *R* có một mảng các biến mệnh đề *Variables* với kích thước của mảng là *k*. Nếu với k biểu thức *Sub_C* = [C_1, C_2,…, C_k] ⊂ *C* nào:

+ *Replace(Variables, Sub_C, R.Left).getID() == p.getID(),*
  thì đặt *p = Replace(Variables, Sub_C, R.Right).*
+ *Replace(Variables, Sub_C, R.Right).getID() == p.getID(),*
  thì đặt *p = Replace(Variables, Sub_C, R.Left).*

3. Cho một biểu thức *p*, *p* có một biểu thức con là *g*. Luật tương đương *R*. Nếu:

+ *R.Left.getID() == g*, thì *p = Replace(g, R.Right, p)*
+ *R.Right.getID() == g*, thì *p = Replace(g, R.Left, p)*

4. Cho một biểu thức *p*, *p* có một biểu thức con *g*, *g* có một mảng các biểu thức con *C*. Luật *R*, *R* có một mảng các biến mệnh đề *Variables* với kích thước của mảng là *k*. Nếu với k biểu thức *Sub_C* = [C_1, C_2,…, C_k] ⊂ *C* nào:

+ *Replace(Variables, Sub_C, R.Left).getID() == g.getID(),*
  thì đặt *p = Replace*(*g, Replace(Variables, Sub_C,R.Right), p)*.
+ *Replace(Variables, Sub_C, R.Right).getID() == g.getID(),*
  thì đặt *p = Replace*(*g, Replace(Variables, Sub_C, R.Left), p)*.

Định nghĩa: Bộ Cheker cải tiến cho một số luật.

Luật	**Dạng Luận lý**
R = Luật phép kéo theo	Check: f.O ==TOANTU.KeoTheo
R = Luật tương đương	Check: f.O = TOANTU.TuonDuong
R = Luật đồng nhất	Check: Contain(f,False) AND f.O = TOANTU.TUYEN

	OR Contain(f,TRUE) AND f.O = TOANTU.HOI
R = Luật nuốt	Check: Contain(f,TRUE) AND f.O = TOANTU.TUYEN **OR** Contain(f,FALSE) AND f.O = TOANTU.HOI

Bảng 4-1 Checker cải tiến của luật tương đương

4.1.2 Sự suy diễn logic

Định nghĩa: Suy diễn lôgic là lập luận mà trong đó kết luận được rút ra từ các sự kiện được biết trước theo kiểu: nếu các tiền đề là đúng thì kết luận phải đúng. Nghĩa là các sự kiện cho trước đòi hỏi rằng kết luận là đúng.

Định nghĩa: *Bộ kiểm tra cho luật suy diễn*

$$Checker_{DR} = Checker_{Default} \cup Checker_{Advance}$$

**Bộ kiểm tra mặc định:*

Giả sử ta có một tập giả thiết GT, một luật suy diễn r có thể áp dụng vào GT nếu hoặc GT có chứa các tiền đề của r, hoặc GT có chứa các biểu thức tương đương với các tiền đề trong r.

**Bộ kiểm tra cải tiến:*

Rule	Check
$(p \rightarrow q) \wedge p \rightarrow q$	- p.O = ToanTu.KeoTheo -và p.Left = q or p.Left ≡ q

(p → q) ∧ ¬q → ¬p	- p.O = ToanTu.KeoTheo -và p.Right = Not(q) or p.Right ≡ Not(q)
(p→q) ∧ (q→r) → (p → r)	-Chuyển về dạng kéo theo. -Tìm trong GT có q sao cho, q chuyển được về dạng kéo theo và p.Right = q.Left
(p ∨ q) ∧¬q → p	-Chuyển p về dạng Tuyển -Kiểm tra GT có chưa NOT(p.C_i) không. Với C_i là các biểu thức con của p.
p∧q→(p ∧ q)	Đây là một luật phụ trợ.
(p ∧ q) →p	-p.ToanTu = ToanTu.Hoi
p→(p∧q)	Không cần quan tâm đến luật này
(p ∨ q) ∧ (¬p ∨ r) → (q ∨ r)	-Chuyển p về dạng Hội -Tìm q ∈ *GT*, q chuyển được về dạng HỘI. Và với bất kì biểu thức con C_i nào của p, kiểm tra Contain (q,Not(C_i)).

Bảng 4-2 Checker cải tiến cho luật suy diễn

4.1.3 Rút gọn mệnh đề

Định nghĩa : cho hai biểu thúc *f* và *g*. Ta nói rằng, *f* "đơn giản hơn" (ký hiệu "<<") *g* nếu:

Length(f) < Length(g)

Trong đó hàm, *Length(p)* đo độ phức tạp của một biểu thức. Độ phức tạp này phụ thuộc vào số lượng và loại toán tử của biểu thức p. Cụ thể như sau:

STT	Toán tử	Độ phức tạp
1	"( )"	0.1
2	¬	0.5
3	∨,∧	1
4	→	2
5	↔	6
6	NONE	0

Bảng 4-3 Độ phức tạp toán tử

Như vậy, với một biểu thức *p,* mảng các biểu thức con c và bảng độ phức tạp *SC_DIFF* như trên thì:

$$\text{Length(p)} = \sum_{0<i<length(c)} SC\_DIFF(c.O)$$

Ví dụ p = ¬x∨y, Length(p) = 1.5.

Định nghĩa : trạng thái "simplest" của biểu thức *f* là một biểu thức tương đương f mà không tồn tại một trạng thái *f'* nào sao cho *Length(f')* < *Length(simplest(f))*

Xác định yêu cầu: Bài toán rút gọn có thể được hiểu như sau. Cho một biểu thức *f,* việc rút gọn *f* chính là việc tìm biểu thức *g* tương đương với *f*, có độ phức tạp là nhỏ nhất.

g ≡ *f and* ∀*h, g* << *h.* Do đó *g* là một simplest của *f.*

Định nghĩa: Một bước *Transformation v* được định nghĩa bằng một cấu trúc như sau:

v=(Expr,Name)

Trong đó: Expr là biểu thức tương đương

Name là luật được áp dụng cho bước trung gian này.

Xây dựng tập luật tương đương cho quá trình rút gọn:

Tất cả luật tương đương cần phải được sắp xếp theo tiêu chí Length(r.Left) < Length(r.Right). Điều đó có nghĩa biểu thức p phải tương đương với r.Left và biến đổi thành r.Right.

Ý tưởng: Cho một biểu thức *f* và một tập các luật tương đương *R* đã được sắp xếp. Thuật toán sẽ tiến hành biến đổi *f* thành các trạng thái *f'* thông qua *r* (*r* thuộc *R*), sao cho ơ mỗi trạng thái độ phức tập của nó phải nhỏ hơn ở trạng thái trước đó. Sau đó, ta lại phải rút gọn các biểu thức con của *f'*, điều này làm *f'* thay đổi. Bước cuối cùng, ta lại tiếp tục chuyển đổi *f'*. Tuy nhiên nếu xuất hiện một trạng thái *f''* nào mà có *Length(f'') > Length(f')*, thì sau k lần chuyển đổi mà độ phức tạp của nó không giảm xuống, thì simplest của *f* chính là *f'*.

Thuật giải:

Input: p- một biểu thức đầu vào, tập luật cải tiến R

Output: g , *Trans* –danh sách các bước chuyển đổi trung gian .

Thực hiện:

B1: Duyệt *p* qua tập luật *R* thông qua các hàm *Checker* để sinh ra trạng thái *p'* của *p*.

B2: Lần lượt duyệt các biểu thức con C_i của *p* thông qua R, để chuyển đổi các *Ci* thành cách *Ci'*. Rồi lặp lại các biểu thức cấp thấp hơn cho đến cấp thấp nhất là các biến mệnh đề.

B3: Trong quá trình duyệt các biểu thức *q*, sinh ra *q',* cập nhật $simplest_q = q'$. Nếu *q* làm xuất hiện các $q'' \neq q'$. Nếu sau k lần chuyển đổi *q'' ,Length(q'')* < Length(q') thì cập nhật $simplest_q = q''$. Cập nhật q= $simplest_q$. Các quá trình chuyển đổi q được lưu lại vào *Trans.*

B4: Lặp lại quá trình trên đến khi $q = p$.

4.1.4 Chứng minh hai biểu thức tương đương

Dựa vào công thức 1.1 . Ta có được nhận xét sau:

Nhận xét:

1. Cho hai biểu thức *f* và *g* . Nếu *g* là *simplest* của *f* thì *g* tương đương *f*.
2. Cho hai biểu thức *f* và *g*. Nếu *simplest(f) = simplest(g)* thì *f* tương đương với *g*. Ngược lại, Nếu *simplest(f) ≠ simplest(g)* thì *f* không tương đương với *g* .

⁎**Xác định vấn đề:** bài toán chúng minh hai mệnh đề tương đương được hiểu như sau. Cho biểu thức T có dạng p ≡ q. Tìm các trạng thái biến đổi P_T của p sao cho p $\xrightarrow{P_T}$ q. Nếu:

1. simplest (p) = q thì P_T = Simplify(p).
2. simplest (q) = p thì P_T = Reverse(Simplify(q))
3. Simplify(p).Lastest = Simplify(q).Lastest thì P_T = (Simplify(p) ∪ Simplify(q)) / (Simplify(p) ∩ Simplify(q))

Thuật giải:

Input: biểu thức T có dạng p ≡ q , tập luật tương đương được cải tiến R

Output: [] nếu p không tương đương q, CD nếu p tương đương q.

Thực hiện:

B1: Tách T thành hai biểu thức p và q.

B2: Đặt T_p = *Simplify(p)* và T_q = *Simplify(q)*.

, $simplest_p$ = Lastest(T_p) và $simplest_q$= Lastest(T_q)

B3: Nếu : $simplest_p$ = q thì đặt CD = T_p

B4: Nếu: $simplest_q$ = p thì đặt CD = Reverse (T_q)

B5: Nếu: $simplest_q$ = $simplest_p$ thì đặt CD = T_p . Duyệt ngược T_q, với mỗi i (0<=i<T_q.length) thì

+ nếu i ≠0, CD.push(t_i.Name , t_{i-1}.BieuThuc)

+ nếu i = 0, CD.push(t_i.Name, q)

B6: Default CD = [].

4.1.5 Suy diễn logic

Xác định vấn đề: bài toán suy diễn logic được biểu diễn như sau. Cho tập các biểu thức giả thiết GT và biểu thức mục tiêu G.Tìm các suy diễn từ các luật R. để chứng minh G ∈ GT.

Đinh nghĩa: một lời giải suy diễn được định nghĩa bởi cấu trúc sau:

Reasoning = (Id, Expr,Parent_Id, Rule)

Trong đó: Id là thứ tự của lời giải, nếu id = 0 thì đó là các giả thiết của đề bài

Expr là kết quả thu được sau bước suy diễn này,

Parent_Id: tập hợp các cơ sở để có được bước suy diễn này

Rule: Luật được áp dụng

Ý tưởng: Ta sẽ dùng suy diễn tiến để thực hiện quá trình suy luận. Lần lượt lấy các thành phần của tập giả thiết, rồi áp dụng các luật đã được định nghĩa cho mệnh đề đang xét đó để tạo ra các sự kiện hay ràng buộc khác. Quá trình này được dừng lại cho đến khi, hoặc không thể tiến hành suy diễn được nữa hoặc đã tìm thấy mục tiêu O từ tập GT.

Phương pháp này có ưu điểm là đơn giản, dễ cài đặt. Tuy nhiên, cách thức này đôi khi sẽ nảy sinh ra các bước dư thừa, khiến không gian tìm kiếm trở lên lớn, gây tốn kém tài nguyên. Đôi khi các giả thiết dư thừa cũng làm cho quá trình suy diễn trở nên phức tạp.

Vì thế, chúng tôi xin được phép đề xuất một giải pháp để hạn chế vấn đề này. Đó là sau mỗi một phép biến đổi hay suy diễn, chúng ta sẽ sắp xếp lại tập giả thiết sao cho nó càng gần mục tiêu O càng tốt. Cụ thể, các giả thiết có các biểu thức sơ cấp và toán tử càng giống với mục tiêu thì sẽ được ưu tiên duyệt trước, các giả thiết còn lại sẽ được xếp phía sau. Những giả thiết là sự kiện sẽ có độ ưu tiên thấp nhất đề hạn chế duyệt chúng. Sau quá trình sắp xếp trên ta lại thực hiện tiếp quá trình biến đổi và suy diễn cho đến khi gặp điều kiện dừng.

Để miêu tả quá trình sắp xếp này, chúng ta xem xét ví dụ. Cho mô hình suy diễn sau:

$$\begin{array}{l} p \vee q \\ \overline{p} \\ \overline{q} \vee r \\ s \rightarrow \overline{r} \\ \hline \therefore \overline{s} \end{array}$$

Ta tiến hành sắp xếp lại tập giả thiết:

Giả thiết	**Loại**	**Số lượng biểu thức sơ cấp giống nhau**	**Điểm**
p ∨ q	Ràng buộc	0	0
¬p	Sự kiện	0	-1
¬q ∨ r	Ràng buộc	0	0

s → ¬r	Ràng buộc	1	1

Bảng 4-4 Giả thiết sau khi được sắp xếp

Tập giả thiết sau khi đươc sắp xếp lại

Giả thiết	Điểm
s → ¬r	1
p ∨ q	0
¬q ∨ r	0
¬p	-1

Truy vết lời giải:

- Sau quá trình suy diễn trên có thể khẳng định được ta có thể suy diễn G từ GT đươc hay không. Nhưng để minh họa cho quá trình suy diễn đó ta cần một tập các Reasoning (*xem lại cấu trúc mục định nghĩa 3.3.2*). Do đó ta cần phải truy vết được tập Reasoning cần thiết.

- Lưu ý rằng: kết thúc quá trình suy diễn ta sẽ nhận được một tập hợp các lời giải *Reasoning*, gọi là *TongQuat*. Ban đầu *TongQuat* chỉ bao gồm các biểu thức giả thiết, với *Id* = 0. Sau mỗi một bước suy diễn, sẽ tạo ra một dòng *Reasoning* t_i của *TongQuat.* Đồng thời ta cần lấy các cơ sở của bước suy diễn này và tìm trong *TongQuat*, nhằm lấy các *Id* và thêm chúng vào trong tập *Parent_Id* của t_i.

- Quá trình truy về sẽ sử dựng bằng phương pháp đệ quy, bắt đầu từ biểu thức mục tiêu G có trong lời *Reasoning* cuối cùng của *TongQuat*. Chúng ta sẽ tiến hành duyệt ngược lên các biểu thức cơ sở của nó qua tập Parent_Id chứa trong mỗi lời giải. Ứng với mỗi Id đó sẽ được tham chiếu đến các lời giải trong *TongQuat*. Quá trình duyệt sẽ dẫn đến lời giải cấp thâp nhất là các Giả thiết (Id=0). Khi đó ta tiến hành PUSH nó vào trong tập *KetQua*.

Kể từ dây, quá trình duyệt sẽ đi hướng ngược lại, khi gặp bất kỳ lời giải tiếp theo nào, nếu nó không có trong *KetQua,* thì thêm nó vào *KetQua* với Id mới, còn nếu nó đã có trong *KetQua* ta chỉ cần lấy lời giải đó ra. Kêt thức quá trình đệ quy, *KET_QUA* là tập đang tìm

- Quá trình truy vết được định nghĩa tại hàm *Truy_Vet_Suy_Dien(TONG_QUAT).*

Thuật giải cho bài toán suy diễn:

Input: Tập giả thiết GT và mục tiêu G

Ouput: Tập KET_QUA

Thực hiện:

B1: Ghi nhận tập giả thiết GT và tập mục tiêu O, cài đặt tập TONG_QUAT với các giả thiết ban đầu.

B2: Kiểm tra O có thuộc GT không nếu có đưa ra kết luận với kết quả = *Truy_Vet_Suy_Dien(TONG_QUAT).*

B3: Sắp xếp lại tập sự kiện theo tiêu chí giống với tập kết luận

B4: Duyệt giả thiết GT_i thuộc tập GT, theo thứ tự sắp xếp, nêu có bất kỳ giả thiết nào có thể áp dụng được luật L_i nào được thì.

- Tạo một lời giải mới S, tìm kiếm các id của các cơ sở và cập nhật vào S. Ghi nhận S vào TONG_QUAT
- Ghi nhận kết quả vào tập GT đồng thời loại bỏ GT_i ra tập GT
- Tiến hành lặp lại bước 2.

B5: nếu đã duyệt qua hết giả thiết mà không còn có thể áp dụng luật nào nữa thì ta tiến hành kết luận mô hinh suy diễn là sai.

4.1.6 Xác định chân trị biểu thức

Xác định yêu cầu: cho một biểu thức mệnh đề p, kết quả trả về là một ma trận 2^MxN .

Với M là số lượng biến mệnh đề cấu thành nó

N là số lượng biểu thức trong p kể cả p.

Thuật giải:

Input: biểu thức p

Output: ma trận MT MxN

Thực hiện:

B1: Duyệt và ghi tất cả các biến mệnh đề có trong mệnh đề P vào trong tập A.

B2: Đếm tất cả các biêu thức có trong P (kể cả P). Và tạo ra ma trận MT kích thước MxN.

B2: Từ tập A, sinh ra tất cả trạng thái ứng với các trường hợp giá trị có thể xảy ra cho các biến mệnh đề, và lưu chúng vào mảng T.

B3: Ứng với từng trạng thái T_i có trong T. Ta tiến hành cập nhật chân trị của các mệnh đề dựa vào giá trị của các biến mệnh đề và các toán tử, theo chiều hướng từ mệnh đề cấp thấp nhất đến mệnh đề cấp cao nhất. Ghi nhận chân trị đó vào MT.

B4: lặp lại bước 3 đến khi T không còn nữa. Kết quả cuối cùng là ma trận MT

4.2 Mô hình giải các bài toán đại số Boolean

Trong miền tri thức đại số Boolean, bài tập chính đó là đơn giản hóa hàm Boolean. Để giải được dạng bài này, chúng ta cần thực hiện bốn quá trình sau:

- Ánh xạ đa thức biểu diễn hàm Boolean lên bìa Karnaugh
- Xác định các tế bào lớn
- Xác định các nhóm tế bào
- Xác định các đa thức tối giản nhất.

4.2.1 Ánh xạ đa thức biểu diễn hàm Boolean lên bìa Karnaugh

Định nghĩa: tế bào trong bìa Karnaugh được biểu diễn bằng một chuỗi nhị phân, nó được tao ra bằng cách liên kết các biến trong đa thức Boolean lại với nhau. Do đó, một hàm Boolean có n biến sẽ có 2^n tê bào trong bìa Karnaugh tương ứng.

Ma trận biểu diễn bìa Karnaugh là một mảng ma trận, do các bài tập trong chương trình dạy ở các trường từ 3-5 biến. Do đó bìa Karnaugh được biểu diễn thông qua một mảng không gian 3 chiều.

Thuật toán:

Input: chuỗi đa thức.

Output: mảng ba chiều được ánh xạ từ đa thức.

Các bước:

Bước 1: Xác định số lượng biến, khởi tạo mảng ba chiều M theo *Quy tắc thứ tự Bìa Karnaugh*

Bước 2: Với mỗi một đơn thức T_i của đa thức F_n , sẽ được chuyển thành một số C_i = ChuyenDoi(T_i). Các số đó được thêm vào mảng A.

Bước 3: Tiến hành duyệt mảng M nếu bất kỳ phần tử nào của M không thuộc A. Ta gáng phần tử đó là -1.

4.2.2 Xác định tế bào lớn.

Biểu diễn tế bào lớn: Do các tế bào được biểu diễn thông qua các số tự nhiên (khác -1). Nên tế bào lớn được biểu diễn lả một mảng các số tự nhiên.

Ý tưởng: Do số lượng các biến của đề bài thường được cố định từ 3-5 biến. Vì thế, ta hoàn toàn xác định được tế bào lớn nhất có thể xuất hiện . Một tế bào lớn có dạng 2^n ô, nên trường hợp khả thi nhất là tế bào 16 ô, tế bào 8 ô,…, tế bào 1 ô. Để có thể lọc dược các tế bào này, ta cần tạo một khung cố định rồi lần lượt duyệt qua bìa Karnaugh. Bộ khung được xây dựng dựa vào số lượng các biến trong bìa Karnaugh, cụ thể:

**Bìa Karnaugh 3 biến:*

-Nhóm tế bào 16 biến: không có.

-Nhóm tế bào 8 biến : không khả thi.

-Nhóm tế bào 4 biến:

	$\overline{A}\overline{B}$	$\overline{A}$B	AB	A$\overline{B}$
$\overline{C}$				
C				

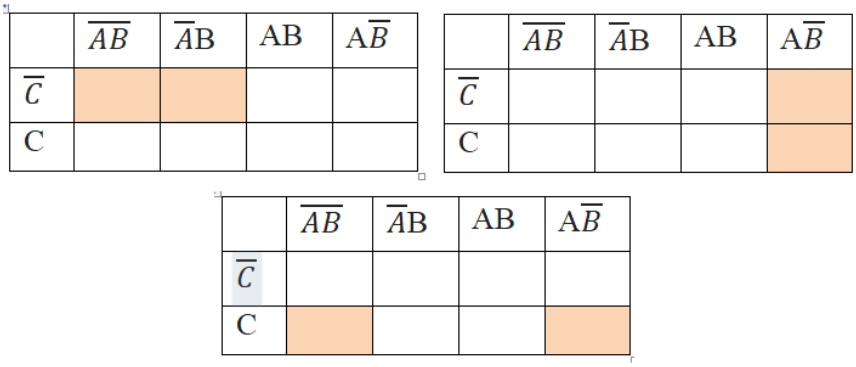
Ô vuông 4

	$\overline{A}\overline{B}$	$\overline{A}$B	AB	A$\overline{B}$
$\overline{C}$				
C				

Đối xứng qua cột

Hình 4-1 Nhóm tế bào 4 của bìa Karnaugh 3 biến

-Nhóm tế bào 2:

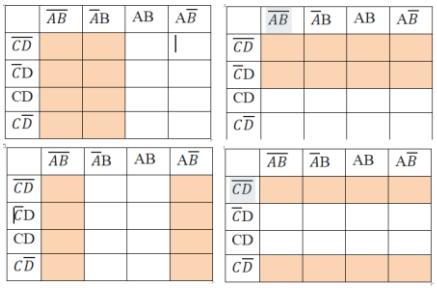


Hình 4-2 Nhóm tế bào 2 của bìa Karnaugh 3 biến

**Bìa Karnaugh 4 biến:*

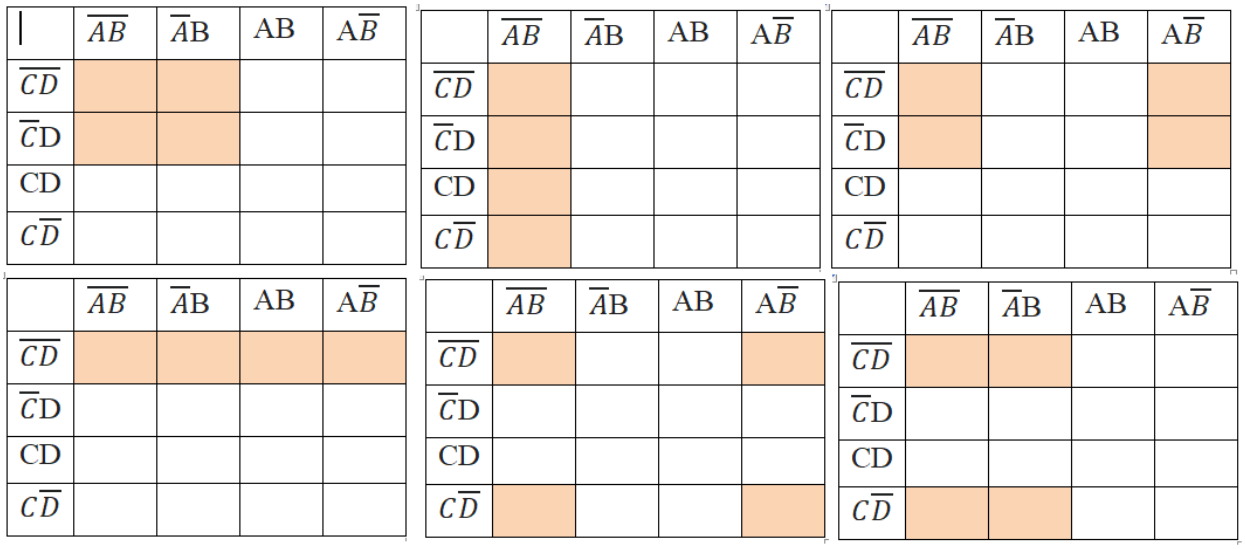
-Nhóm tế bào 16 ô: không thả khi.

-Nhóm tế bào 8 ô:



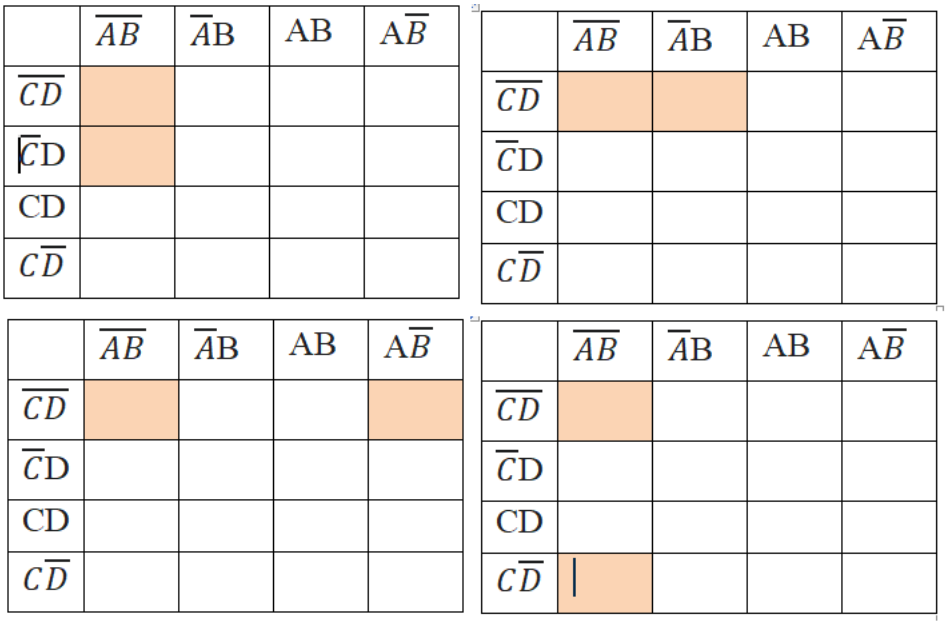
Hình 4-3 Nhóm tế bào 8 của bìa Karnaugh 4 biến

-Nhóm tế bào 4:



Hình 4-4 Nhóm tế bào 4 bìa Karnaugh 4 biến

-Nhóm tế bào 2



Hình 4-5 Nhóm tế bào 3 của biafKarnaugh 4 biến

4.2.3 Xác định các nhóm tế bào lớn

Sau khi đã ánh xạ hàm Boolean lên một bìa Karnaugh, bước tiếp theo ta cần tìm các nhóm tế bào. Sao cho toàn bộ các phần tử của mõi nhóm tế bào vừa bao phủ hết kar(f).

Thuât giải như sau:

-Đầu tiên, duyệt qua ma trận biểu diễn bìa Karnnaugh, thống kê ở từng ô có bao nhiêu tế bào đã bao phủ nó.

-Sau khi thống kê hoàn thành. Với các ô chỉ có một tế bào được bao phủ, tế bào đó chính là một tế bào thiết yếu và ta cần lưu giữ nó. Đồng thời với tế bào này, ta phải duyệt và đánh dấu cho các ô mà tế bào này chứa đựng.

-Sau khi hoàn thành bước trên, nếu vẫn còn xót một ô nào trong Kar(f), thì ô đó được trên hai tế bào lớn bao phủ. Trường hợp này ta tiến hành lấy từng trường hợp của các tế bào này. Từ đó tìm được tất cả các đa thức của f.

4.2.4 Xác định đa thức tối giản.

Định nghĩa: *hai biểu thức đơn giản hơn.*

Cho hai công thức đa thức của một hàm Bool :F = m1+m2 +…. +m_k và G =M1 + M2 +… + M_l.Ta nói rằng công thức F đơn giản hơn công thức G nếu tồn tại đơn ánh h: {1,2,..,k} → { 1,2,…, l} sao cho với mọi I ∈ {1,2,..,k} thì số từ đơn của m_i không nhiều hơn số từ đơn của $M_{h(i)}$.

Sau khi tìm được tất cả đa thức của f. Dựa vào *định nghĩa hai biểu thức đơn giản hơn*, đa thức f được xem là tối giản nhất nếu f có số đơn thức là ít nhất.

4.3 Xây dựng mô hình quan hệ hai ngôi.

Các bài toán của quan hệ hai ngôi bao gồm: xác định các tính chất của quan hệ, chứng minh quan hệ tương đương và chứng minh quan hệ thứ tự.

4.3.1 Xác định các tính chất của quan hệ

Đối với bài toán trên quan hệ R có cấu trúc danh sách phần tử, được hiểu như sau. Cho R, R có một tập các phần tử *elements*. Với mỗi một tập luật $r_i \in$ Rules, thì $R.tinhChat_i = r_i(R)$.

Các luật xác định tính chất được quy định trong **mục 3.3**

4.3.2 Xác minh quan hệ tương đương.

Đối với bài toán này ta cần chia ra các trường hợp cụ thể:

**Tập không gian cụ thể - Quan hệ cụ thể*

Xét một quan hệ R có tập không gian A. Nếu R và A được biểu diễn bằng danh sách phần tử. Xác định một nhóm các lớp tương đương phan hoạch A như sau:

Bước 1: Ánh xạ R thành một ma trận M.

Bước 2: Tạo một mảng label chỉ định nhãn của từng thành phần trong A. Đặt index = 0.

Bước 3: Nêu label[index] là rỗng thì đặt label[index] = index.

Bước 4: Tiến hành duyệt trên ma trận M. Nếu có bất kỳ $M_{index\ j}$ nào khác 0, label[j] là rỗng và index ≠ j. Thì ta gáng label[j] = index, và đặt index = j rồi lặp lại bước 2.

Bước 5: tiến hành duyệt lại label để xem phần tử nào chưa xet nhãn để đặt index là thành phần đó rồi lại tiếp tục lặp bước 2.

Bước 6: các phần tử trong A đã được dánh nhãn chính là các lớp cần tìm.

**Tập không gian cụ thể - Quan hệ đặc tả*

Với một R là dạng đặc tả nhưng có tập không gian mẫu A xác định hữu hạn các phần tử. Ta tiến hành chuyển R thành dạng cấu trúc danh sách phần tử. Bằng cách duyệt từng phần tử trong A, rồi xét với các phần tử khác có trong A, xem thõa mãn điều kiện không. Sau đó, chuyển thành trường hợp *Tập không gian cụ thể - Quan hệ cụ thể.*

**Tập không gian đặc tả - Quan hệ đặc tả*

Với trường hợp này hệ thống chỉ hỗ trợ dạng *quan hệ đại số* chia hết và số chia là một số nguyên. Giả sử số nguyên đó là m, để tìm các lớp tương ứng của R, ta chỉ cần tìm các số a sau cho, a mod m = k. (k từ 0 đến m-1). Do đó nó m lớp tương đương.

4.3.3 Xác minh quan hệ tương đương.

Bài toán chỉ xét trong trường hợp A là một tập hợp có hữu hạn các phần tử. Việc giải quyết bài toán xác minh quan hệ tương đương sẽ xoay quanh đối tượng biểu đồ Hasse.

Định nghĩa: *Biểu diễn cấu trúc biểu đồ Hasse.*

1. Một đỉnh của biểu đồ Hasse gọi là một node, mỗi node có giá trị bằng một số và một danh sách các node cha(là các node trội của nó) và node con (là các node bị trội của nó).
2. Hasse được biểu diễn bằng một ma trận, với các node không có cha gọi là các seed. Các node không có con gọi là các leaf.

Xây dựng Hasse được thực hiện theo quy tắc sau:

B1: Ánh xạ R thành một ma trận M.

B2: Nếu Hase đang rỗng ta khởi tạo một node tương đương phần tử trên A. Rồi gáng nó vào *seed* trong Hasse.

B3: Duyệt Hasse từ trên xuống nếu:

+ Giá trị i hiện tại, mà có bất kỳ một j nào trong *Hasse* làm $M_{ij} = 1$ thì ta khởi tạo một node i nằm tầng trên node j và gáng cha của I là j, con của j là i.

+ Nếu không có bất kỳ phần tử nào Hasse có quan hệ với i thì gáng i vào *seed* của *Hasse*

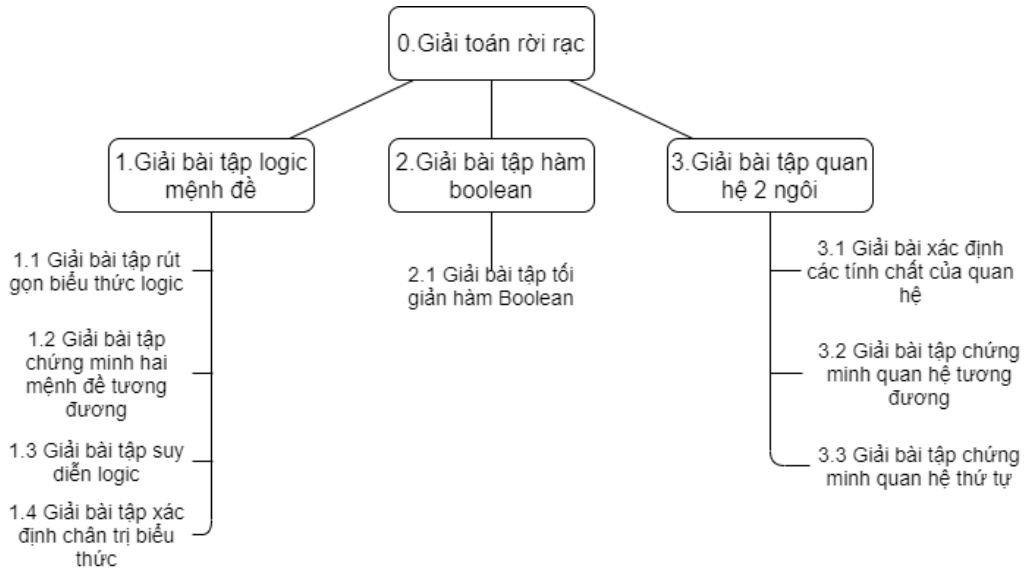
Đối với các thuộc tính khác của quan hệ thứ tự thì:

- Tối tiểu: là tập seed của Hasse.
- Tối đại là tập các leaf của Hasse.
- Giá trị nhỏ nhất: khi seed có một giá trị duy nhất thì nó chính là giá trị nhỏ nhất.
- Giá trị lớn nhất: khi tập leaf có duy nhất một phần tử thì nó là giá trị lớn nhất.

CHƯƠNG 5. HỆ THỐNG GIẢI TOÁN RỜI RẠC

Hệ thống giải toán rời rạc được xây dựng thành một website đơn giản. Chương trình hỗ trợ giải các bài tập theo từng bước, phù hợp cho những ai đang có mong muốn tham khảm.

5.1 Phân tích nghiệp vụ hệ thống



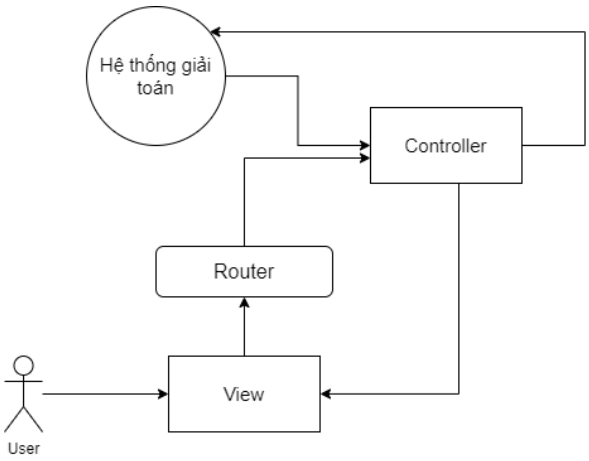
Sơ đồ 5-1 Sơ đồ nghiệp vụ của hệ thống

Hệ thống được xây dựng có các chức năng chính như sau:

- Giải bài tập liên quan logic mệnh đề: người dùng tiến hành nhập chuỗi biểu thức. Hệ thống tiến hành đưa ra lời giải theo đúng yêu cầu của người dùng như: rút gọn biểu thức, chứng minh hai mệnh đề tương đương nhau, suy diễn logic và xuất ra chân trị của một biểu thức.
- Giải bài tập hàm Boolean: chỉ có duy nhất một bài tập được hỗ trợ, đó là tối giản hàm Boolean. Đầu vào là chuỗi đa thức hoặc danh sách các ô của Kar(f). Lời giải là tất cả các đa thức tối giản của f.
- Giải tập chương quan hệ hai ngôi: xác định các tính chất của một quan hệ, chứng minh và tìm các thuộc tính của quan hệ hế tương đương. Chứng minh và tìm các tính chất của quan hệ thứ tự. Lưu ý: các quan hệ này chỉ nằm trong tập số nguyên Z.

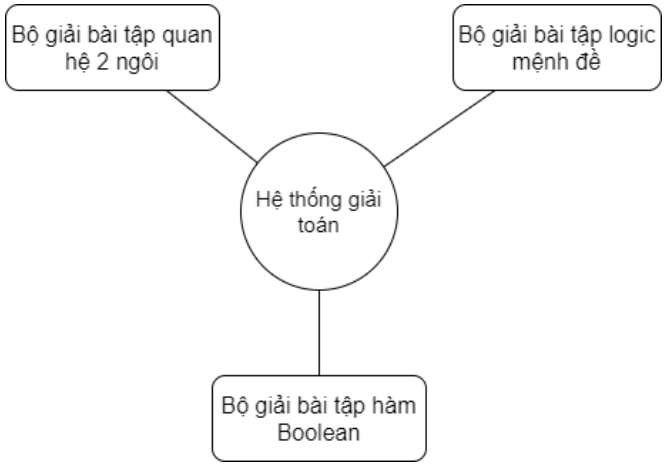
5.2 Cấu trúc tổng quát của hệ thống

Hệ thống được xây dựng thành website có cấu trúc như sau:



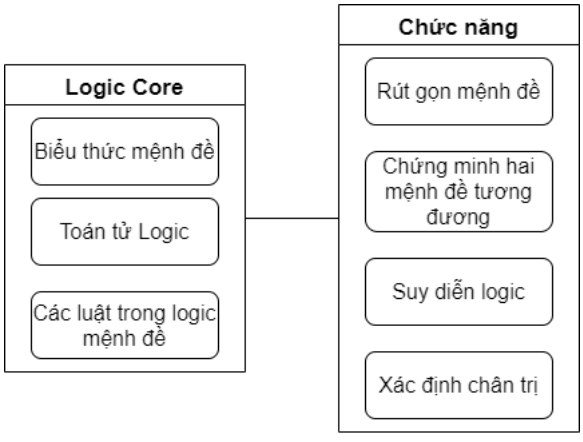
Sơ đồ 5-2 Cấu trúc website giải toán rời rạc

Trong đó cấu trúc cốt lõi, hệ thống giải toán bao gồm:



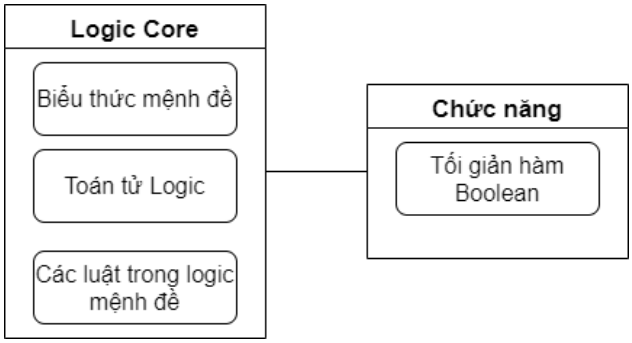
Sơ đồ 5-3 Cấu trúc hệ thống giải toán

**Cấu trúc bộ giải bài tập logic mệnh đề:* Cấu trúc này được ánh xạ từ cấu trúc của mô hình về miền tri thức Logic mệnh đề đã được đề cập trước đó:



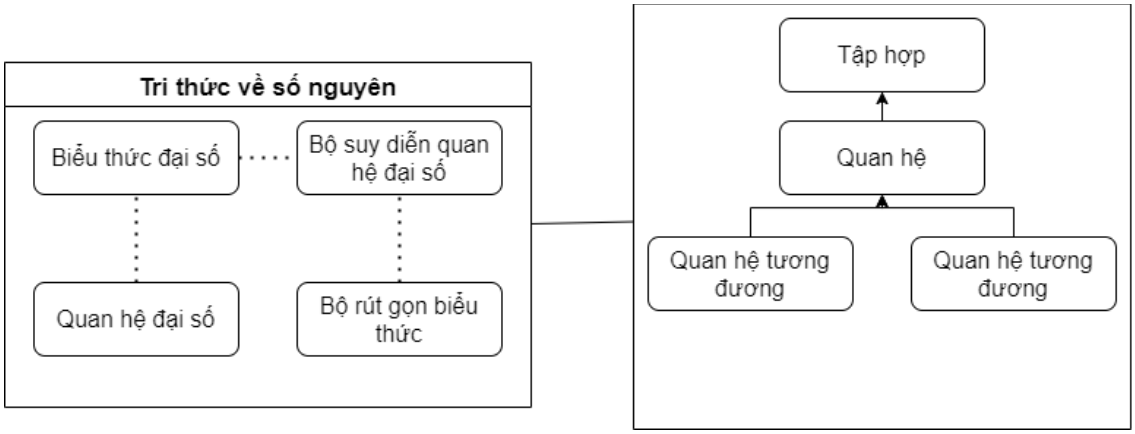
Sơ đồ 5-4 Cấu trúc bộ giải bài tập logic mệnh đề

**Cấu trúc bộ giải bài tập hàm Boolean:*



Sơ đồ 5-5 Cấu trúc bộ giải bài tập hàm Boolean

**Cấu trúc bộ giải bài tập quan hệ hai ngôi:*các thành phần chính bao gồm.



Sơ đồ 5-6 Cấu trúc bộ giải bài tập quan hệ hai ngôi.

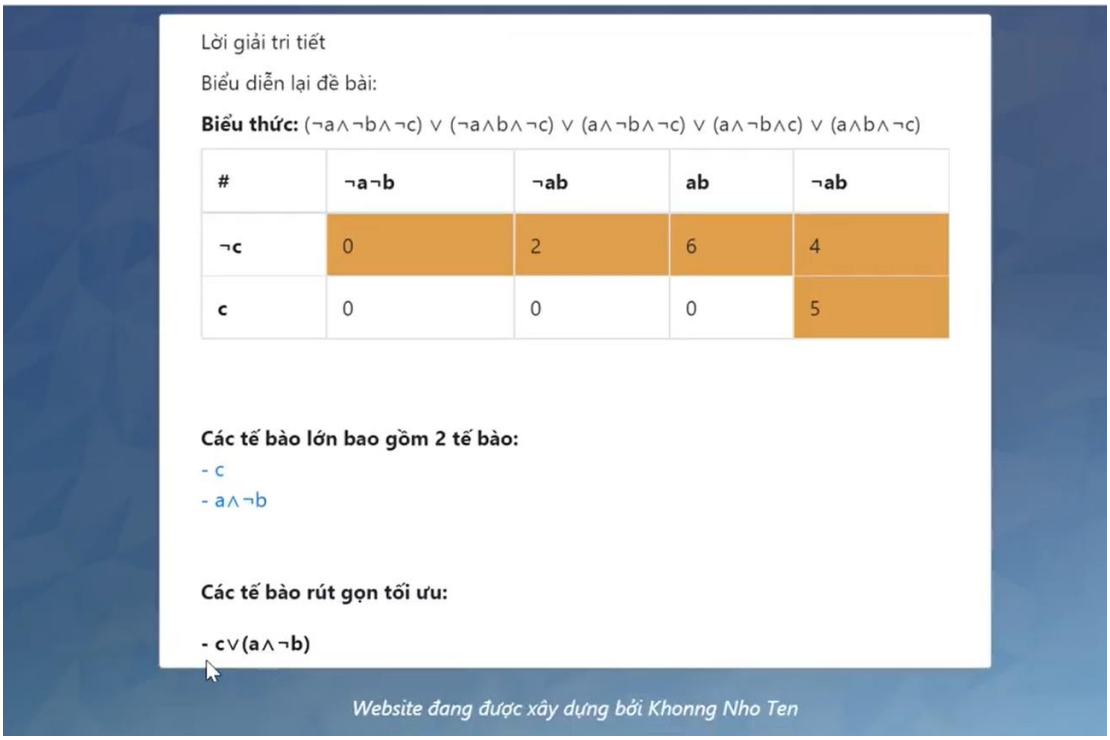
5.3 Giao diện chương trình

Hình 5-1 Màng hình nhập liệu

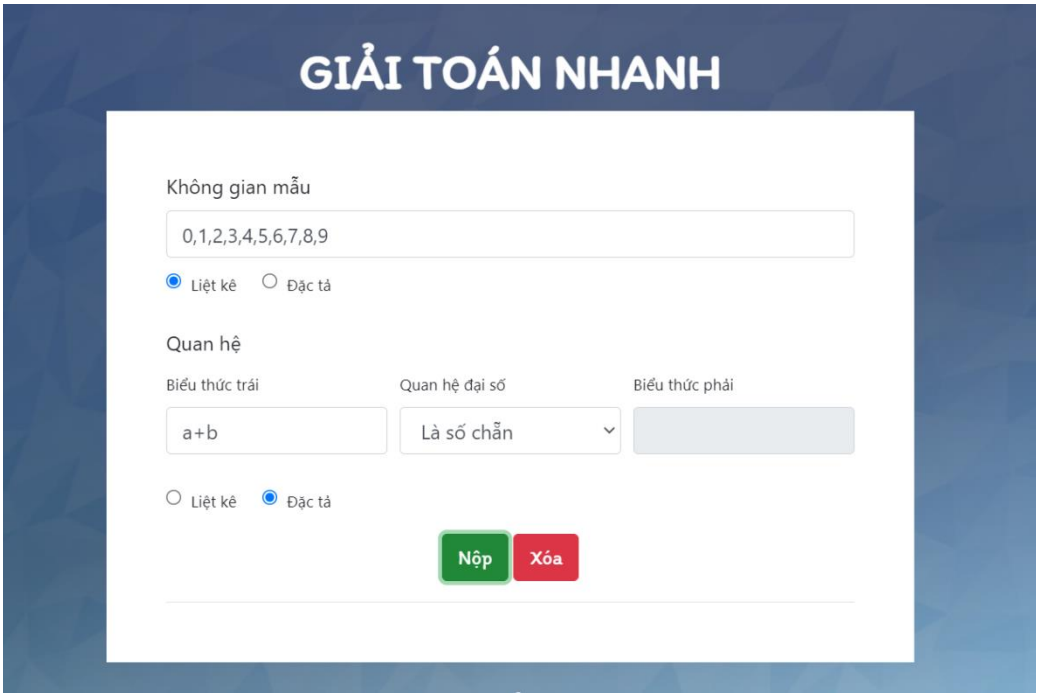
Hình 5-2 Màng hinh suy diễn logic

Hình 5-3 Màng hình xác định chân trị

Hình 5-4 Màng hình chứng minnh hai mệnh đề tương đương



Hình 5-5 Màng hình bài toán rút gọn hàm Boolean



Hình 5-6 Màng hình bài toán quan hệ hai ngôi

5.4 Công nghệ sử dụng

5.4.1 Nodejs

Bảng 5-1 Logo nodejs

NodeJS là một nền tảng được xây dựng trên V8 JavaScript Engine – trình thông dịch thực thi mã JavaScript, giúp xây dựng các ứng dụng web một cách đơn giản và dễ dàng mở rộng.

NodeJS được phát triển bởi Ryan Dahl vào năm 2009 và có thể chạy trên nhiều hệ điều hành khác nhau: OS X, Microsoft Windows, Linux.

Nodejs được ứng dụng trong các việc xây dựng:

- Websocket server: Các máy chủ web socket như là Online Chat, Game Server…

- Fast File Upload Client: là các chương trình upload file tốc độ cao.
- Ad Server: Các máy chủ quảng cáo.
- Cloud Services: Các dịch vụ đám mây.
- RESTful API: đây là những ứng dụng mà được sử dụng cho các ứng dụng khác thông qua API.
- Any Real-time Data Application: bất kỳ một ứng dụng nào có yêu cầu về tốc độ thời gian thực.
- Micro Services: Ý tưởng của micro services là chia nhỏ một ứng dụng lớn thành các dịch vụ nhỏ và kết nối chúng lại với nhau.

5.4.2 Typescript

Hình 5-7 Logo typescript

Typescript là một dự án mã nguồn mở được Microsoft phát triển, được xem là một phiên bản nâng cao của Javascript. TypeScript là một ngôn ngữ giúp cung cấp quy mô lớn hơn so với JavaScript.

Vì sao lại được xem là phiên bản nâng cao của Javascript? Vì nó được bổ sung những tùy chọn kiểu tĩnh và các lớp hướng đối tượng, nó bao hàm luôn ES6(ECMAScript 6 2015) - phiên bản mới nhất của Javascript.

TypeScript thêm các namespace, class và module tùy chọn vào JavaScript. TypeScript hỗ trợ các công cụ cho các ứng dụng JavaScript quy mô lớn cho bất kỳ trình duyệt nào, cho bất kỳ máy chủ nào, trên bất kỳ hệ điều hành nào.

5.5 Kiểm thử

Sau khi đã xây dựng thành công website giải toán rời rạc. Em đã tiến hành giải một số bài toán và có được một số thống kê sau đây:

Loại bài	**Số lượng**	**Thành công**

Rút gọn biểu thức logic	32	29
Chứng minh hai biểu thức tương đương	42	38
Suy diễn logic	41	36
Xác định chân trị	21	21
Tối giản hàm Boolean	24	24
Xác định tính chất quan hệ	31	28
Chứng minh quan hệ tương đương	28	25
Chứng minh quan hệ thứ tự	22	18

Bảng 5-2 Thông kê kết quả

CHƯƠNG 6. TỔNG KẾT

6.1 Kết quả đạt được

Qua bài báo cáo này, em đã trình bày được về lý thuyết của mô hình COKB đồng thời cải tiến nó để phù hợp với miền tri thức Toán rời rạc. Qua đó đã xây dựng được thành công môt hệ thống có thể hỗ trợ việc giải các bài tập toán rời rạc một cách dễ hiểu theo từng bước.

6.2 Vấn đề tồn tại

- Cấu trúc mô hình chưa hoàn thiện
- Hệ thông còn một số lỗi
- Giao diện người dùng còn xấu.
- Chưa hoàn thiện quá trình kiểm tra đầu vào.

6.3 Kiến nghị

- Hoàn thiện mô hình biểu diễn.
- Khắc phục các lỗi của hệ thống web
- Cải thiện giao diện người dùng
- Nâng cấp lên các nền tảng khác nhất là di động.
- Thêm chức năng tra cứu kiến thức.

PHỤ LỤC

Phụ lục 1: hướng dẫn sử dụng

- Demo: https://toanroiracnhanh.herokuapp.com/
- Source code: https://github.com/KhongTaiKhoan/toanroiracnhanh
- Hoặc truy cập https://www.youtube.com/channel/UCl5h9rYAGlEFKXMbeLKJX-g để xem demo qua video.

Phụ lục 2: yêu cầu nhập liệu

- Yêu cầu nhập biểu thức mệnh đề thì:
  + Cần phải có dấu "( )" để phân tác các biểu thức con.
  + True là 1 và False là 0.
  + Không được để các ký tự đặc biệt.
  + Chấp nhận khoảng trắng.
- Yêu cầu nhập quan hệ hai ngôi:
  + Chỉ cho phép trong phạm vi số nguyên.
  + Không được nhập sai biểu thức điều kiện

TÀI LIỆU THAM KHẢO

[1] Kenneth H. Rosen, Discrete Mathematics and Its applications, 7th edition, McGraw-Hill (2012).

[2] C. Li, K. Mehrotra, Problems on Discrete Mathematics, Syracuse University, New York (2011).

[3] "Final Report on the 2013 NSF Workshop on Research Challenges and Opportunities in Knowledge

Representation" Natasha Noy and Deborah McGuinness (Eds). National Science Foundation Workshop

Report, August 2013.

[4] William F. Klostermeyer, Discrete Mathematics Problems, University of North Florida (2012).

[5] Do, V.N., 2012. Intelligent Problem Solvers in Education: Design Method and Applications, Intelligent Systems, In: Vladimir M. Koleshko (Ed.), Intelligent systems, InTech, pp. 121-148.

[6] Aladova, E., Plotkin, T., 2017. Logically automorphically equivalent knowledge bases, arXiv:1707.01027v1

[7] Do, V.N., 2015. Ontology COKB for knowledge representation and reasoning in designing knowledge-based systems. Communications in Computer and Information Science (CCIS), vol. 513, Springer, pp. 101-118.

[8] Do, V.N., Nguyen, D.H., Mai, T.T., 2015. Reasoning Method on Knowledge about Functions and Operators, International Journal of Advanced Computer Science and Applications (IJACSA),

6(6): 156 – 168.

[9] Hien D. Nguyen and Nhon V. Do, Intelligent Problem Solver in Education for

Discrete Mathematics University of Information Technology, New Trends in Intelligent Software Methodologies, Tools and Techniques (SoMeT_17)

[10] Tổng quan về Nodejs https://trungquandev.com/mot-cai-nhin-tong-quan-nhat-ve-nodejs/

[11] Giới thiệu về typescript https://viblo.asia/p/gioi-thieu-typescript-su-khac-nhau-giua-typescript-va-javascript-LzD5dDn05jY