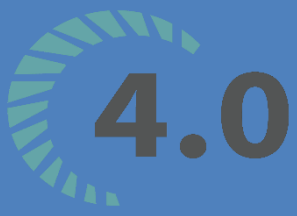
KHOA CÔNG NGHỆ THÔNG TIN

ĐẠI HỌC KHOA HỌC TỰ NHIÊN THÀNH PHỐ HỒ CHÍ MINH, ĐẠI HỌC QUỐC GIA TP HCM

CƠ SỞ TRÍ TUỆ NHÂN TẠO



Sinh viên thực hiện: Nguyễn Quang Hiển

Mã số sinh viên: 20120077

Giảng viên phụ trách: Nguyễn Duy Khánh - Nguyễn Ngọc Băng Tâm

Đồ án/Bài tập: Logic

ĐỒ ÁN/BÀI TẬP MÔN HỌC - CƠ SỞ DỮ TRÍ TUỆ NHÂN TẠO

HỌC KỲ I – NĂM HỌC 2022-2023

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
227 Nguyễn Văn Cừ, Phường 4, Quận 5, TP.HCM
Điện Thoại: (08) 38.354.266 - Fax:(08) 38.350.096

4.0

YÊU CẦU ĐỒ ÁN - BÀI TẬP

Loại bài tập	☐ Lý thuyết ☑ Thực hành ☑ Đồ án ☑ Bài tập
Ngày bắt đầu	21/11/2022
Ngày kết thúc	05/12/2022

Hợp giải trên logic mệnh đề

Mục lục

1. Hợp giải

Hệ thống hợp giải là một hệ thống chứng minh đúng và đủ đối với bài toán suy dẫn. Hệ thống này dựa trên một luật duy nhất được gọi là luật hợp giải được phát biểu như sau: cho các mệnh đề A, B, C khi đó:

$$\frac{A \vee \bar{B}}{\frac{B \vee \bar{C}}{A \vee C}}$$

hay $(A \vee \bar{B}) \wedge (B \vee \bar{C}) \Longrightarrow (A \vee C)$

Luật hợp giải trên là đúng vì nếu A đúng thì rõ ràng kết luận trên là đúng và luật là đúng. Ngược lại nếu A sai thì B cũng phải sai (do $A \vee \bar{B}$) và do đó C là đúng vì $(B \vee C)$ đúng.

1.1. Dạng hội chuẩn (CNF)

Luật hợp giải chỉ áp dụng được lên các câu nối rời của các từ, do đó nó chỉ thích hợp với các cơ sở tri thức và truy vấn chứa các phép hội như trên. Một câu logic được biểu diễn nối liền là các phép hội của từ được gõi là dạng hội chuẩn (*Conjunctive Normal Form*) hay CNF.

Mọi câu logic mệnh đề đều tương đương logic với một câu dạng hội chuẩn. Thủ tục biến đổi gồm 4 bước:

Bước 1: Loại bỏ $\Leftrightarrow$, thay $\alpha \Leftrightarrow \beta$ bằng $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$

Bước 2: Loại bỏ $\Rightarrow$, thay $\alpha \Rightarrow \beta$ bằng $\bar{\alpha} \vee \beta$

Bước 3: CNF đòi hỏi dấu phủ định chỉ được xuất hiện trên các từ, do đó cần phân phối dấu phủ định vào bên trong (các luật De Morgan)

Bước 4: Áp dụng luật phân phối $\vee$ vào $\wedge$ cho tất cả trường hợp có thể

Ví dụ: với câu $A \Leftrightarrow (B \vee C)$, kết quả các bước biến đổi lần lượt là:

Bước 1: $(A \Rightarrow (B \vee C)) \wedge ((B \vee C) \Rightarrow A)$

Bước 2: $(\bar{A} \vee B \vee C) \wedge ((\overline{B \vee C}) \vee A)$

Bước 3: $(\bar{A} \vee B \vee C) \wedge ((\bar{B} \wedge \bar{C}) \vee A)$

Bước 4: $(\bar{A} \vee B \vee C) \wedge (\bar{B} \vee A) \wedge (\bar{C} \vee A)$

Một số luật biến đổi tương đương trong mệnh đề logic:

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha && \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{De Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{De Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}
$$

Figure 7.11 Standard logical equivalences. The symbols α, β, and γ stand for arbitrary sentences of propositional logic.

Ngữ nghĩa dạng hội chuẩn:

$$
\begin{aligned}
\mathit{CNFSentence} &\rightarrow \mathit{Clause}_1 \wedge \cdots \wedge \mathit{Clause}_n \\
\mathit{Clause} &\rightarrow \mathit{Literal}_1 \vee \cdots \vee \mathit{Literal}_m \\
\mathit{Fact} &\rightarrow \mathit{Symbol} \\
\mathit{Literal} &\rightarrow \mathit{Symbol} \mid \neg \mathit{Symbol} \\
\mathit{Symbol} &\rightarrow P \mid Q \mid R \mid \ldots \\
\mathit{HornClauseForm} &\rightarrow \mathit{DefiniteClauseForm} \mid \mathit{GoalClauseForm} \\
\mathit{DefiniteClauseForm} &\rightarrow \mathit{Fact} \mid (\mathit{Symbol}_1 \wedge \cdots \wedge \mathit{Symbol}_l) \Rightarrow \mathit{Symbol} \\
\mathit{GoalClauseForm} &\rightarrow (\mathit{Symbol}_1 \wedge \cdots \wedge \mathit{Symbol}_l) \Rightarrow \mathit{False}
\end{aligned}
$$

Figure 7.12 A grammar for conjunctive normal form, Horn clauses, and definite clauses. A CNF clause such as $\neg A \vee \neg B \vee C$ can be written in definite clause form as $A \wedge B \Rightarrow C$.

1.2. Giải thuật hợp giải (Robinson)

Giải thuật hợp giải sẽ được trình bày trong hình phí dưới. Đầu tiên ($KB \wedge \bar{\alpha}$) được biến đổi về dạng CNF. Sau đó, hợp giải được áp dụng lên các mệnh đề kết quả. Mỗi cặp mệnh đề chứa các từ bù nhau được hợp giải để tạo ra mệnh đề mới và được thêm vào tập câu nếu nó chưa xuất hiện. Quá trình tiếp tục đến khi một trong 2 điều kiện sau xảy ra:

- Không có mệnh đề mới nào có thể thêm vào, trong trường hợp đó KB không suy dẫn được α

- Hai mệnh đề hợp giải nhận được một mệnh đề rỗng, thì KB suy dẫn được α

Mã giả thuật toán hợp giải:

function PL-RESOLUTION(KB, α) **returns** *true* or *false*
  inputs: KB, the knowledge base, a sentence in propositional logic
    α, the query, a sentence in propositional logic

  clauses ← the set of clauses in the CNF representation of $KB \wedge \neg\alpha$
  new ← { }
  while *true* **do**
    for each pair of clauses C_i, C_j **in** *clauses* **do**
      resolvents ← PL-RESOLVE(C_i, C_j)
      if *resolvents* contains the empty clause **then return** *true*
      new ← *new* ∪ *resolvents*
    if *new* ⊆ *clauses* **then return** *false*
    clauses ← *clauses* ∪ *new*

Figure 7.13 A simple resolution algorithm for propositional logic. PL-RESOLVE returns the set of all possible clauses obtained by resolving its two inputs.

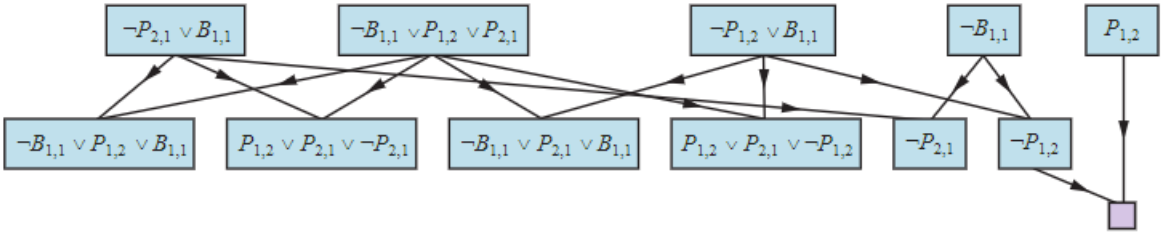


Figure 7.14 Partial application of PL-RESOLUTION to a simple inference in the wumpus world to prove the query $\neg P_{1,2}$. Each of the leftmost four clauses in the top row is paired with each of the other three, and the resolution rule is applied to yield the clauses on the bottom row. We see that the third and fourth clauses on the top row combine to yield the clause $\neg P_{1,2}$, which is then resolved with $P_{1,2}$ to yield the empty clause, meaning that the query is proven.

1.3. Đánh giá giải thuật hợp giải

- ***Ưu điểm:***
  - Độ hoàn thiện: Nếu một tập hợp các mệnh đề là không thỏa mãn, thì kết thúc quá trình hợp giải của các mệnh đề đó chứa mệnh đề trống.
  - Độ đúng đắn: Kết quả trả về luôn là hệ quả được suy ra từ cơ sở tri thức ban đầu.
- ***Nhược điểm:***

- Mỗi cặp mệnh đề hợp giải với nhau có thể tạo ra nhiều kết quả (có nhiều cặp từ bù giữa hai câu) và các kết quả không góp phần vào quá trình chứng minh đúng
- Hệ thống đòi hỏi vét cạn sẽ duyệt qua hết các trường hợp hợp giải gây mất nhiều thời gian và không khả thi

- ***Giải pháp:***
  - Đối với giải thuật hợp giải (Robinson) thì sử dụng trong trường hợp các mệnh đề có ít literal
  - Sử dụng một số giải thuật cải tiến **DPLL** hay **WalkSET**
  - Sử dụng giải thuật **DPLL** (backtracking search):
    - Early termination
    - Pure symbol heuristic
    - Unit clause heuristic

    Mã giả:

```
function DPLL-SATISFIABLE?(s) returns true or false
  inputs: s, a sentence in propositional logic

  clauses ← the set of clauses in the CNF representation of s
  symbols ← a list of the proposition symbols in s
  return DPLL(clauses, symbols, { })

function DPLL(clauses, symbols, model) returns true or false

  if every clause in clauses is true in model then return true
  if some clause in clauses is false in model then return false
  P, value ← FIND-PURE-SYMBOL(symbols, clauses, model)
  if P is non-null then return DPLL(clauses, symbols – P, model ∪ {P=value})
  P, value ← FIND-UNIT-CLAUSE(clauses, model)
  if P is non-null then return DPLL(clauses, symbols – P, model ∪ {P=value})
  P ← FIRST(symbols); rest ← REST(symbols)
  return DPLL(clauses, rest, model ∪ {P=true}) or
         DPLL(clauses, rest, model ∪ {P=false})
```

  - Sử dụng giải thuật **WalkSAT** (local hill-climbing search):
    - *Mã giả:*

```
function WALKSAT(clauses, p, max_flips) returns a satisfying model or failure
  inputs: clauses, a set of clauses in propositional logic
          p, the probability of choosing to do a "random walk" move, typically around 0.5
          max_flips, number of value flips allowed before giving up

  model ← a random assignment of true/false to the symbols in clauses
  for each i = 1 to max_flips do
    if model satisfies clauses then return model
    clause ← a randomly selected clause from clauses that is false in model
    if RANDOM(0, 1) ≤ p then
      flip the value in model of a randomly selected symbol from clause
    else flip whichever symbol in clause maximizes the number of satisfied clauses
  return failure
```

2. Kịch bản kiểm thử

Xây dựng 5 kịch bản kiểm thử với 3 kịch trả về kết quả là "YES" và 2 kịch bản trả về kết quả là "NO"

2.1. Kịch bản 1

input0.txt	output0.txt	Ghi chú
-A	3	
4	-A	(-A OR B) hợp giải với (-B)
-A OR B	B	(-A OR B) hợp giải với (A)
B OR C	-C	(-C OR B) hợp giải với (-B)
A OR -B OR C	4	
-B	-B OR C	(A OR -B OR C) hợp giải với (-A)
	A OR C	(A OR -B OR C) hợp giải với (B)
	A OR -B	(A OR -B OR C) hợp giải với (-C)
	{}	(-B) hợp giải với (B)
	YES	**KB entails α vì tồn tại mệnh đề rỗng trong KB**

2.2. Kịch bản 2

input1.txt	output1.txt	Ghi chú
A	2	
4	-C	(-A OR B) hợp giải (-B)
-A OR B	-B OR C	(A OR C OR -B) hợp giải (-A)
-C OR B	2	
A OR C OR -B	A OR -B	(A OR C OR -B) hợp giải (-C)
-B	-A OR C	(-A OR B) hợp giải (-B OR C)

	1	
	A OR -C	(A OR -B) hợp giải (-C OR B)
	0	
	NO	**KB không entails α vì không phát sinh được mệnh đề mới và không tìm thấy mệnh đề rỗng**

2.3. Kịch bản 3

input2.txt	output2.txt	Ghi chú
A	3	
4	A	(-B OR A) hợp giải (B)
-A OR B OR C	-C	(-C OR A) hợp giải (-A)
-B OR A	-B	(-B OR A) hợp giải (-A)
-C OR A	4	
B	B OR C	(-A OR B OR C) hợp giải (A)
	-A OR C	(-A OR B OR C) hợp giải (-B)
	{}	(-A) hợp giải (A)
	-A OR B	(-A OR B OR C) hợp giải (-C)
	YES	**KB entails α vì tồn tại mệnh đề rỗng trong KB**

2.4. Kịch bản 4

input3.txt	output3.txt	Ghi chú
-C	4	
3	A	(A OR -C) hợp giải (C)
-A OR B	B OR -C	(-A OR B) hợp giải (A OR -C)
-B OR -C	-A OR -C	(-A OR B) hợp giải (-B OR -C)
A OR -C	-B	(-B OR -C) hợp giải (C)
	3	
	-C	(-A OR -C) hợp giải (A)
	B	(B OR -C) hợp giải (C)
	-A	(-A OR -C) hợp giải (C)
	1	
	{}	(-A) hợp giải (A)
	YES	**KB entails α vì tồn tại mệnh đề rỗng trong KB**

2.5. Kịch bản 5

input4.txt	output4.txt	Ghi chú
A OR -B	3	
3	A	(A OR -B) hợp giải (B)
-A OR B OR C	-B	(A OR -B) hợp giải (-A)
A OR -B	-C	(A OR -C) hợp giải (-A)
A OR -C	4	
	-A OR C	(-A OR B OR C) hợp giải (-B)
	-A OR B	(-A OR B OR C) hợp giải (-C)
	B OR C	(-A OR B OR C) hợp giải (A)
	{}	(-B) hợp giải (B)
	YES	**KB entails α vì tồn tại mệnh đề rỗng trong KB**

3. Tiêu chí tự đánh giá

STT	Đặc tả tiêu chí	Điểm	Điểm đánh giá
1	Đọc dữ liệu vào và lưu trong cấu trúc dữ liệu phù hợp	0.5	0.5
2	Cài đặt giải thuật trên logic mệnh đề	1	1
3	Các bước suy diễn phát sinh đề và kết luận đúng	2.5	2.5
4	Tuân thủ mô tả định dạng của đề bài	0.5	0.5
5	Báo cáo test case và đánh giá	0.5	0.25
	Tổng	**5**	**4.75**

4. Tài liệu tham khảo

[1] Artificial Intelligence A Modern Approach (4th Edition)

[2] AI Slide for Education - Nguyễn Ngọc Đức

[3] AI/PL Resolution - Cheng Lin Li

[4] Logic - Arizona University