Die Logik des Suppression Effects

Fritz Hamm
IMS Universität Stuttgart

5. Juli 2007

Struktur des Vortrags

- Byrnes Daten zum Modus Ponens
- Planning und Logikprogrammierung
- Syntax und Semantik von normalen Logikprogrammen
- Eine logische Form für Konditionale
- Anwendungen 1. Teil
- Integritätsbedingungen??
- Anwendungen 2. Teil??

Reasoning to an interpretation versus reasoning from an interpretation

(1) If there is an emergency then you press the alarm button. The driver will stop if any part of the train is in a station.

(2) The driver will stop the train if someone presses the alarm button and any part of the train is in a station.

Modus Ponens

(3) 95 %

a. If she has an essay to write she will study late in the library.
b. She has an essay to write.
c. She will study late in the library.

(4) 38%

a. If she has an essay to write she will study late in the library.
b. If the library stays open then she will study late in the library.
c. She has an essay to write.
d. She will study late in the library.

Alternative premisses

(5) 90 %

a. If she has an essay to write she will study late in the library.
b. If she has some textbooks to read, she will study late in the library.
c. She has an essay to write.
d. She will study late in the library.

Other fallacies

(6) AC 71%

a. If she has an essay to write she will study late in the library.
b. She will study late in the library.
c. She has an essay to write.

(7) DA 49%

a. If she has an essay to write she will study late in the library.
b. She doesn't have an essay to write.
c. She will not study late in the library.

(8) MT 69 %

a. If she has an essay to write, she will study late in the library.

b. She will not study late in the library.

c. She does not have an essay to write

Additional premisses

(9) AC 54%

a. If she has an essay to write she will study late in the library.
b. If the library stays open then she will study late in the library.
c. She will study late in the library.
d. She has an essay to write.

(10) DA 22%

a. If she has an essay to write she will study late in the library.
b. If the library stays open then she will study late in the library.
c. She doesn't have an essay to write.
d. She will not study late in the library.

(11) MT 44 %

a. If she has an essay to write, she will study late in the library.
b. If the library stays open then she will study late in the library.
c. She will not study late in the library.
d. She does not have an essay to write

Alternative premisses

(12) AC 16 %

a. If she has an essay to write she will study late in the library.
b. If she has some textbooks to read, she will study late in the library.
c. She will study late in the Library.
d. She has an essay to write.

(13) DA 4 %

a. If she has an essay to write she will study late in the library.
b. If she has some textbooks to read, she will study late in the library.
c. She does not have an essay to write.
d. She will not study late in the library.

(14) DA 22%

a. If she has an essay to write, she will study late in the library.
b. If the library stays open then she will study late in the library.
c. She hasn’t an essay to write.
d. She will study late in the library.

(15) MT 69 %

a. If she has an essay to write, she will study late in the library.
b. If she has a textbook to read, she will study late in the library.
c. She will not study late in the library.
d. She does not have an essay to write

Planning is defined as setting a goal and devising a sequence of actions that will achieve that goal, taking into account events in, and properties of the world and the agent.

goal G can be achieved in circumstances C

goal G can be achieved in circumstances C + D

Definition

A *positive clause* is a formula of the form

$$p_1, \dots p_n \rightarrow q,$$

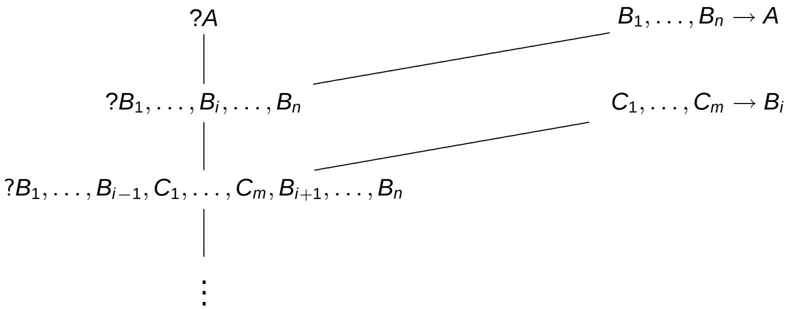
where the q, p_i are propositional variables; the antecedent may be empty.
In this formula, q is called the *head*, and $p_1, \dots p_n$ the *body* of the clause.
A *positive program* is a finite set of positive clauses.

Definition

A *query* is a finite (possibly empty) sequence of atomic formulae denoted as $?p_1, \dots, p_m$. Alternatively, a query is called a *goal*. The empty query, canonically denoted by $\Box$, is interpreted as $\bot$, i.e. a contradiction.

Definition

Unit-resolution is a derivation rule which takes as input a program clause $p_1, \ldots p_n \to q$ and a query $?q$ and produces the query $?p_1, \ldots p_n$.



An illustration of a derivation with unit resolution

Definition

Let P be a positive program on a finite set of proposition letters L. An assignment $\mathcal{M}$ of truthvalues $\{0, 1\}$ to L is a *model* of P if for $q \in L$,

1. $\mathcal{M}(q) = 1$ if there is a clause $p_1, \ldots p_n \rightarrow q$ in P such that for all i, $\mathcal{M}(p_i) = 1$
2. $\mathcal{M}(q) = 0$ if for all clauses $p_1, \ldots p_n \rightarrow q$ in P there is some p_i for which $\mathcal{M}(p_i) = 0$.

Theorem

Let P be a positive program, A an atomic formula. Then $P \models A$ if and only if the empty query can be derived from $?A$ using P.

Definition

Let P be a positive program.

a. The *completion* of a positive program P is given by the following procedure:

1. take all clauses $\varphi_i \rightarrow q$ whose head is q and form the expression $\bigvee_i \varphi_i \rightarrow q$
2. if q does not occur as a head, introduce the clause $\bot \rightarrow q$
3. replace the implications ($\rightarrow$) by bi-implications ($\leftrightarrow$) .
4. take the conjunction of the (finitely many) sentences thus obtained; this gives the completion of P, which will be denoted by $comp(P)$.

b. If P is a positive logic program, define the non-monotonic consequence relation $\approx$ by

$$P \approx \varphi \text{ iff } comp(P) \models \varphi.$$

- $p \rightarrow q$
- p

Constructing models

Definition

The operator T_P associated to P transforms an assignment $\mathcal{V}$ (identified with the set of proposition letters made true true by $\mathcal{V}$) into a model $T_P(\mathcal{V})$ according to the following stipulations: if u is a proposition letter,

1. $T_P(\mathcal{V})(u) = 1$ if there exists a set of proposition letters C, made true by $\mathcal{V}$, such that $\bigwedge C \rightarrow u \in P$
2. $T_P(\mathcal{M})(u) = 0$ otherwise.

Definition

An ordering $\subseteq$ on assignments $\mathcal{V}, \mathcal{W}$ is given by: $\mathcal{V} \subseteq \mathcal{W}$ if all proposition letters true in $\mathcal{V}$ are true in $\mathcal{W}$.

Lemma

If P is a positive logic program, T_P is monotone in the sense that $\mathcal{V} \subseteq \mathcal{W}$ implies $T_P(\mathcal{V}) \subseteq T_P(\mathcal{W})$.

Definition

A fixed point of T_P is an assignment $\mathcal{V}$ such that $T_P(\mathcal{V}) = \mathcal{V}$.

Lemma

If T_P is monotone, it has a least and a greatest fixed point.

Definition

a. A (definite) clause is a formula of the form
$(\neg)p_1 \wedge \ldots \wedge (\neg)p_n \rightarrow q$,
where the p_i are either propositional variables, $\top$ or $\bot$ and q is a propositional variable.

b. A definite logic programm $\mathcal{P}$ is a conjunction of definite clauses.

Problem: negation in the antecedent

$\neg p \rightarrow p$

Strong Kleene

p	$\neg p$
1	0
0	1
u	u

p	q	$p \wedge q$
1	1	1
0	0	0
u	u	u
1	0	0
1	u	u
0	1	0
0	u	0
u	1	u
u	0	0

p	q	$p \vee q$
1	1	1
0	0	0
u	u	u
1	0	1
1	u	1
0	1	1
0	u	u
u	1	1
u	0	u

p	q	$p \rightarrow q$
1	1	1
0	0	1
u	u	u
1	0	0
1	u	u
0	1	1
0	u	1
u	1	1
u	0	u

ŁUKASIEWICZ

p	$\neg p$
1	0
0	1
u	u

p	q	$p \wedge q$
1	1	1
0	0	0
u	u	u
1	0	0
1	u	u
0	1	0
0	u	0
u	1	u
u	0	0

p	q	$p \vee q$
1	1	1
0	0	0
u	u	u
1	0	1
1	u	1
0	1	1
0	u	u
u	1	1
u	0	u

p	q	$p \supset q$
1	1	1
0	0	1
u	u	1
1	0	0
1	u	u
0	1	1
0	u	1
u	1	1
u	0	u

Definition

A three-valued model is an assignment of the truth values $u, 0, 1$ to the set of proposition letters. If the assignment does not use the value u, the model is called *two-valued*. If $\mathcal{M}, \mathcal{N}$ are models, the relation $\mathcal{M} \leq \mathcal{N}$ means that the truth value of a proposition letter p in $\mathcal{M}$ is less than or equal to the truth value of p in $\mathcal{N}$ in the canonical ordering on $u, 0, 1$.

Definition

The completion of a definite logic program comp($\mathcal{P}$) is defined by the following clauses:

a. If a propositional variable p does not occur in the consequent of a clause, add a formula $\bot \rightarrow p$.

b. If a formula is of the form q, i.e. the consequent of a clause with empty antecedent, add a formula $\top \rightarrow q$.

c. For each propositional variable q, collect the clauses $\phi_i \rightarrow p$ with q as consequent, form $\bigvee_i \phi_i$ and add the formula $\bigvee_i \phi_i \leftrightarrow q$.

If P is a normal logic program, define the non-monotonic consequence relation $\approx$ by

$$P \approx_3 \varphi \textit{ iff } comp(P) \models_3 \varphi.$$

More generally if S is a set of atoms occurring in P, the completion of P relatvized to S, $comp_S(P)$, is obtained by taking the conjunctions of the definitions of the atoms q which are in S.

Example

$P = \{\bot \to p, p \wedge \neg ab \to q\}$
$comp_{\{ab,p\}} = \{\bot \leftrightarrow p, \bot \leftrightarrow ab, p \wedge \neg ab \to q\}$
$comp_{\{ab,p,q\}} = \{\bot \leftrightarrow p, \bot \leftrightarrow ab, p \wedge \neg ab \leftrightarrow q\}$

$$comp_{\{ab,p,q\}}(P) \models_3 \neg q$$

Definition

Let P be a definite program.

a. The operator $\mathcal{T}_P$ applied to formulae constructed using only $\neg$, $\wedge$ and $\vee$ is determined by the above truth tables.

b. Given a three-valued model $\mathcal{M}$, $T_P(\mathcal{M})$ is the model determined by

   0.1 $T_P(\mathcal{M})(q) = 1$ iff there is a clause $\varphi \rightarrow q$ such that $\mathcal{M} \models \varphi$

   0.2 $T_P(\mathcal{M})(q) = 0$ iff there is a clause $\varphi \rightarrow q$ in P and for all clauses $\varphi \rightarrow q$ in P, $\mathcal{M} \models \neg\varphi$

   0.3 $T_P(\mathcal{M})(q) = u$, otherwise

Lemma

If P is a definite logic program, T_P is monotone in the sense that $\mathcal{M} \leq \mathcal{N}$ implies $T_P(\mathcal{M}) \leq T_P(\mathcal{N})$.

Lemma

Let P be a program.

- a. *$\mathcal{M}$ is a model of the $comp(P)$ iff it is a fixed point of T_P.*
- b. *The least fixed point of T_P exists and is reached in finitely many steps ($n + 1$ if the program consists of n clauses). The least fixed point of $\mathcal{T}_P$ will be called the minimal model of P.*

Theorem

Let a definite program P be given, and let A be an atomic formula.

1. *There is a successful derivation starting from ?A if and only if $P \mathrel{|}\!\!\approx_3 A$.*
2. *The query ?A fails finitely if and only if $P \mathrel{|}\!\!\approx_3 \neg A$.*

Example

$P = \{p \rightarrow q, \bot \rightarrow p\}$
$M_1(p) = T_P(M_0)(p) = 0, M_1(q) = T_P(M_0)(q) = u$
$M_2(p) = T_P(M_1)(p) = 0, M_2(q) = T_P(M_1(q)) = 0$

1. $\mathcal{L}$ a formal language into which $\mathcal{N}$ is translated
2. the expression in $\mathcal{L}$ which translates an expression in $\mathcal{N}$
3. the semantics $\mathcal{S}$ for $\mathcal{L}$
4. the definition of validity of arguments $\psi_1, \ldots, \psi_n / \phi$, with premisses ψ_i and conclusion ϕ.

(16) If a glass is dropped on a hard surface, it will break.

(17) If a body is dropped, its velocity will increase as gt^2.

Representation of *If A then B*

If A, and nothing abnormal is the case, then B

$A \wedge \neg ab \rightarrow B$

Definition

In the following, the term program will refer to a finite set of conditionals of the form $A_1 \wedge \ldots \wedge A - n \wedge \neg ab \rightarrow B$, together with the clauses $\bot \rightarrow ab$ for all proposition letters of the form ab occurring in the conditionals. Here, the A_i are proposition letters or negations thereof, and B is a proposition letter. We allow the A_i to be $\top$ or $\bot$.

Example

(18) a. If she has an essay to write she will study late in the library.
b. She has an essay to write.

p = She has an essay to write.
q = She will study late in the library.

(19) a. $p \wedge \neg ab \rightarrow q$
b. p.
c. $\bot \rightarrow ab$

Completion of program (19): $\{p, p \leftrightarrow q\}$

Example

(20)
a. $p \wedge \neg ab \rightarrow q$
b. $r \wedge \neg ab' \rightarrow q$
c. $\neg r \rightarrow ab$
d. $\neg p \rightarrow ab'$
e. p.

r = The library stays open.

(21)
a. $\neg ab' \leftrightarrow p$
b. $\neg ab \leftrightarrow r$

Completion of program (20): $\{p \wedge r \leftrightarrow q, p\}$

Example

(22)
a. $p \wedge \neg ab \rightarrow q$
b. $r \wedge \neg ab' \rightarrow q$
c. p.
d. $\bot \rightarrow ab$
e. $\bot \rightarrow ab'$

r = She has a textbook to read.

Completion of program (22): $\{p, (p \vee r) \leftrightarrow q\}$

Example

(23) a. If she has an essay to write she will study late in the library.
b. She doesn't have an essay to write.
c. She will not study late in the library.

(24) a. $p \wedge \neg ab \rightarrow q$
b. $\neg p$
c. $\bot \rightarrow ab$

Completion of program (23): $\{\neg p, p \leftrightarrow q\}$

Integrity constraints: Motivation

Given q and

$\phi_1 \rightarrow q$

$\phi_2 \rightarrow q$

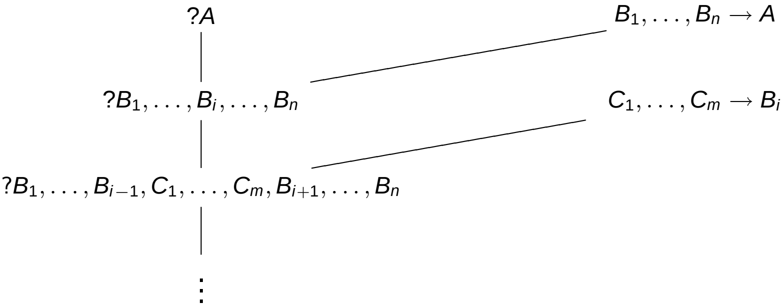
$\vdots$

$\phi_n \rightarrow q$

we want to conclude that q can only be the case because one of the ϕ_i is the case. The notion of completion does not achieve this.

$P = \{p \rightarrow q, \top \rightarrow q\}$

$comp(P) = \{(p \vee \top) \leftrightarrow q\}$



An illustration of a derivation with unit resolution

Definition

$?\phi$ *succeeds* means that a given program P must be transformed via a suitable update into a program P' such that $P' \approx_3 \phi$.

Definition

A conditional integrity constraint of the form

$$\textit{if } ?\psi \textit{ succeeds}, \textit{ then } ?\phi \textit{ succeeds}$$

means: If a program P is given and P' any extension of P such that $P' \approx_3 \psi$, then also $P' \approx_3 \phi$.

Example

(25) AC

a. If she has an essay to write she will study late in the library.
b. She will study late in the library.
c. She has an essay to write.

if $?q$ *succeeds*, *then* $?p$ *succeeds*

Use rule: $p \wedge \neg ab \rightarrow q$

Example

(26) MT

a. If she has an essay to write she will study late in the library.
b. She will not study late in the library.
c. She does not have an essay to write.

if $?q$ *fails*, *then* $?p$ *fails*.

Assumption: $?q$ must fail
Then at least one of $?p$ and $?\neg ab$ must fail
By negation as failure applied to *ab* we get that $?p$ fails

Subject 7
Either she has a very short essay to write or no essay. Maybe she had to write an essay but didn’t ffel like going to the library, or that it was so short that she finished it on time, so she didn’t have to stay.

Example

AC for an additional premiss

$p \wedge \neg ab \rightarrow q$
$r \wedge \neg ab' \rightarrow q$
$\neg p \rightarrow ab'$
$\neg r \rightarrow ab$

Assumption: $?q$ succeeds
Either $?p \wedge \neg ab$ or $?r \wedge \neg ab'$ must succeed
But: $\neg r \leftrightarrow ab$ and $\neg p \leftrightarrow ab'$
Conclusion: Both $?p$ and $?r$ must succeed

Example

MT for an additional premiss

Assumption: $?q$ must fail

The same reasoning as above shows that at least one of p, r must fail. But we don't know which one.