Konditionale und die Logik des “Suppression Effekts”

Fritz Hamm

12. Juli 2010

Überblick

Struktur

- 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

(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.

Die Daten von Byrne: Modus Ponens I

(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.

Die Daten von Byrne: Modus Ponens II

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.

Die Daten von Byrne: Affirmation of the Consequent

(6) 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.

Die Daten von Byrne: Denial of the Antecedent

(7) 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.

Die Daten von Byrne: Modus Tollens

(8) 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

Die Daten von Byrne: Additional Premisses I

(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.

Die Daten von Byrne: Additional Premisses II

(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

Die Daten von Byrne: Alternative Premisses I

(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.

Die Daten von Byrne: Alternative Premisses II

(14) 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

Logic Programming: Positive Programs I

Definition

A *positive clause* is a formula of the form

$$p_1, \ldots p_n \longrightarrow q,$$

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

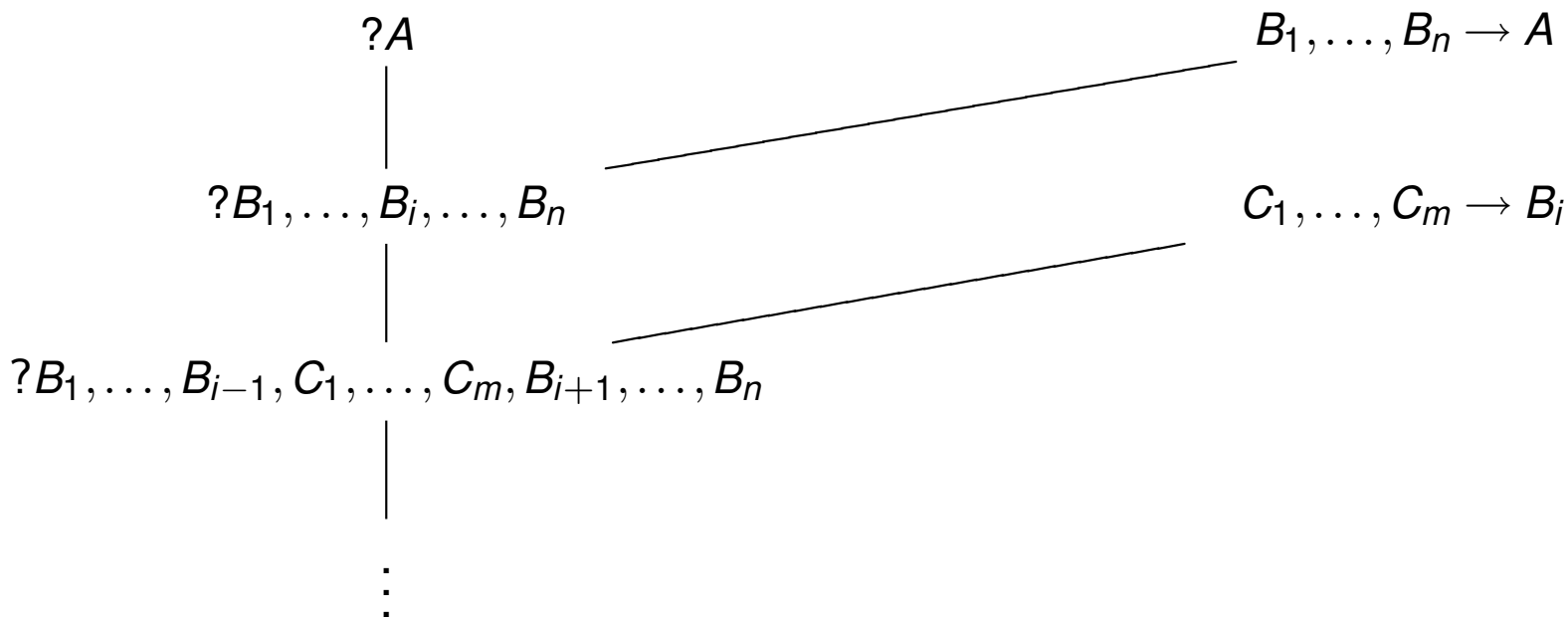
Definition

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

Logic Programming: Positive Programs II

Definition

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



An illustration of a derivation with unit resolution

Logic Programming: Positive Programs III

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.

Logic Programming: Positive Programs IV

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

$$P \,|\!\approx \varphi \textit{ iff } comp(P) \models \varphi.$$

Logic Programming: Positive Programs V

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

$$Comp(P) = \{\top \leftrightarrow p, p \leftrightarrow q\}$$

Constructing Models I

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.

Constructing Models II

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.

Logic Programming: Negation I

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

$$\neg p \rightarrow p$$

Logic Programming: Negation II

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

p	q	$p \leftrightarrow q$
1	1	1
0	0	1
u	u	u
1	0	0
1	u	u
0	1	0
0	u	u
u	1	u
u	0	u

Logic Programming: Negation III

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

p	q	$p \equiv q$
1	1	1
0	0	1
u	u	1
1	0	0
1	u	u
0	1	0
0	u	u
u	1	u
u	0	u

Logic Programming: Negation IV

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 $u, 0, 1$.

Completion and Fixpoints I

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

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

Completion and Fixpoints II

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

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

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

Completion and Fixpoints III

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

Completion and Fixpoints IV

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 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.*

Completion and Fixpoints V

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 $P \mid\!\approx_3 A$.*
2. *The query $?A$ fails finitely if and only if $P \mid\!\approx_3 \neg A$.*

Example

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

Conditionals: Formalization I

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.

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

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

Conditionals: Formalization II

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 $\bot$.

Application I

Example

(17) 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.

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

Completion of program (18):$\{p, p \leftrightarrow q\}$

Application II

Example

(19)
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.

(20)
a. $\neg ab' \leftrightarrow p$
b. $\neg ab \leftrightarrow r$

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

Tutorial Dialogues: Andrea Lechler I

S: Ok yeah I think it is likely that she stays late in the library tonight, but it depends if the library is open . . . so perhaps I think [pauses], yeah, in a way I think hmm what does it say to me? I mean the fact that you first say that she has an essay to write then she stays late in the library, but then you add to it if the library stays open she stays late in the library so perhaps she's not actually in the library tonight, because the library's not open. I don't think it's a very good way of putting it.

E: How would you put it?

S: I would say, if Marian has an essay to write, and the library stays open late, then she does stay late in the library.

Tutorial Dialogues: Andrea Lechler II

E: Again three sentences: If the green light is on, the rabbit is in the cage. If the door is closed, the rabbit is in the cage. I can see that the green light is on. So imagine that' a conversation, you hear these sentences, what can you conclude now?

S: What I conclude? If the cage door is shut, then the rabbit is in the cage. Again, you've met one of the two criteria for the rabbit being in the cage. The green light is on, the rabbit is in the cage. Um, but we don't know whether or not the door is open or not. If the door's shut, then that's it, we got it for sure, the rabbit is definitely in the cage.

E: So you are not sure now?

S: I' am not sure because I don't know whether the door is open or not. The point about the door being open or not hasn't been cleared up to my satisfaction. The point about the green light has been cleared up to my satisfaction, but not the point whether the door is open or not. If the door is open, I'm not sure the rabbit's in the cage, it's probably somewhere else at the time.

Tutorial Dialogues: Andrea Lechler III

S: The second (conditional) again has no function. If you suppose that it is always the case that when she has an essay to write, she studies late in the library, then of course she now stays late in the library.

Application Continued I

Example (Alternative Premisses)

(21)
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 (21): $\{p, (p \vee r) \leftrightarrow q\}$

Application Continued II

Example (Denial of the Antecedent)

(22)
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.

(23)
a. $p \wedge \neg ab \rightarrow q$
b. $\neg p$
c. $\bot \rightarrow ab$

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

Backward Reasoning: Integrity Constraints I

Given q and

$\phi_1 \rightarrow q$

$\phi_2 \rightarrow q$

$\vdots$

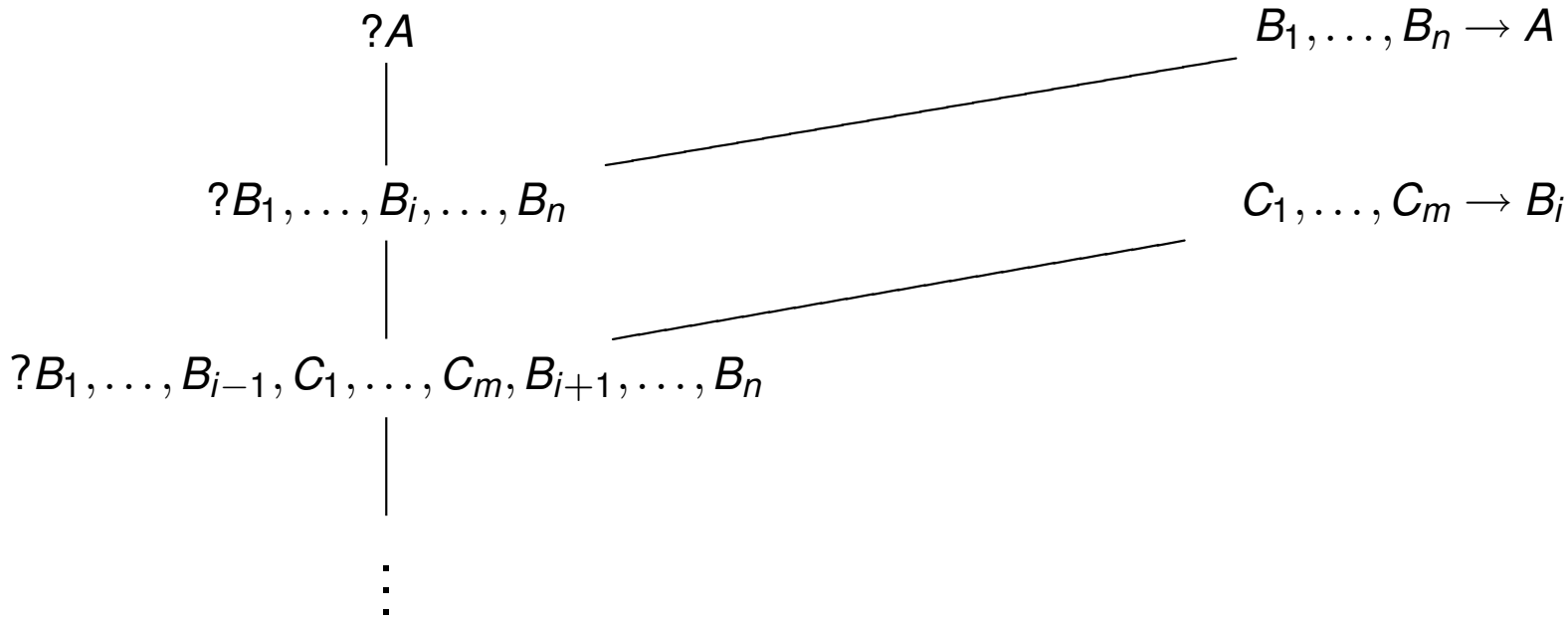
$\phi_n \rightarrow q$

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.

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

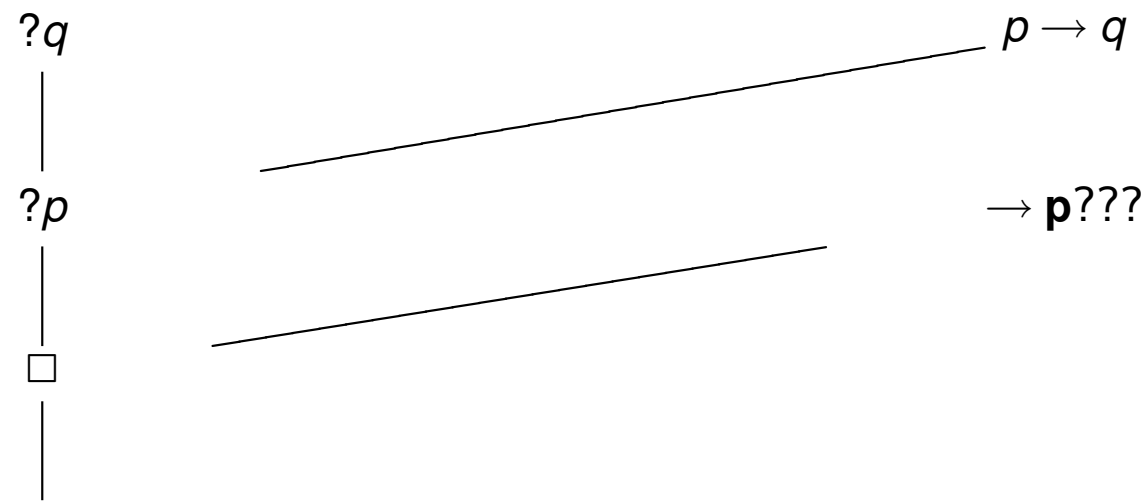
$comp(P) = \{(p \vee \top) \leftrightarrow q\}$

Backward Reasoning: Integrity Constraints II



An illustration of a derivation with unit resolution

Backward Reasoning: Integrity Constraints III



Backward Reasoning: Integrity Constraints IV

Definition

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

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 $P' \mid\!\approx_3 \psi$, then also $P' \mid\!\approx_3 \phi$.

Backward Reasoning II: Application I

Example (Affirmation of the Consequent)

(24)
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: $p \wedge \neg ab \rightarrow q$

Backward Reasoning II: Application II

Example (Modus Tollens)

(25)
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

Backward Reasoning II: Application III

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

Backward Reasoning II: Application IV

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.