Logic and Logical Inference

- Recall basic concepts of logic
- Logical inference
  - » deduction
  - » *abduction*
  - » *induction*
- Clausal Logic
- Deductive Inference (e.g. resolution)
- Abductive Inference
  - » semantics
  - » algorithm

Logic: another thing that penguins aren't very good at.

Logic (a recap)

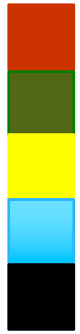
- Humans capable of manipulating logical information and making logical inference

  The red block is on the green block.
  The green block is somewhere above the blue block.
  The green block is not on the blue block.
  The yellow block is on the green block or the blue block.
  There is some block on the black block.

  Facts

  There can be only one block on another.
  A block cannot be two colors at once.

  Background knowledge



- Logic is a mechanism for using and studying valid logical inference.

  $on(red, green) \wedge \neg on(green, blue)$
  $\exists X\ [\ block(X) \wedge on(green, X) \wedge on(X, blue)\ ]$
  $on(yellow, green) \vee on(yellow, blue)$
  $\exists X\ [\ block(X) \wedge on(X, black)\ ]$

  $block(red) \wedge block(yellow) \wedge block(blue) \wedge block(back) \wedge block(green)$
  $\forall X,Y,Z\ [\ on(X, Y) \wedge on(Z, Y) \rightarrow X = Z)\ ]$

Logic (a recap)

Propositional Logic

» propositional variables $p, q, r, s, \ldots..$

» connectives $\neg, \wedge, \vee, \rightarrow$

Syntax

sentences $((p \wedge q) \vee r) \rightarrow (p \wedge r))$

» interpretation assigns each propositional variable a unique true value. Interpretation of sentences is constructed from a given interpretation and truth tables.

$p^i = T, \; q^i = F, r^i = T$ $((p \wedge q) \vee r) \rightarrow (p \wedge r))^i = T$

» logical entailment of a sentence from a set of sentences, given as premises, is when the sentence is true in all interpretations that satisfy the given premises.

$$\{p, p \rightarrow q\} \models q$$

Logic (a recap)

Predicate Logic

» propositional letters — raining, snowing, wet…..

» constants — table, block1, block2, etc.

» variables — X, X_1, Y, Y_1, etc.

» functions — size, color, etc.

» predicates — on, above, clear, block, etc.

Terms

sentences

$\neg$block(table)

$\forall$X block(X) $\leftrightarrow$ X= block1 $\vee$ X=block2 $\vee$ X= block3

$\forall$X,Y (block(X) $\wedge$ block(Y) $\wedge$ size(X) = size(Y) $\rightarrow$ sameSize(X,Y))

$\forall$X clear(X) $\leftrightarrow$ (block(X) $\wedge$ $\neg\exists$Y on(Y, X))

$\forall$X,Y (on(X,Y) $\leftrightarrow$ (block(X) $\wedge$ block(Y)) $\vee$Y= table)

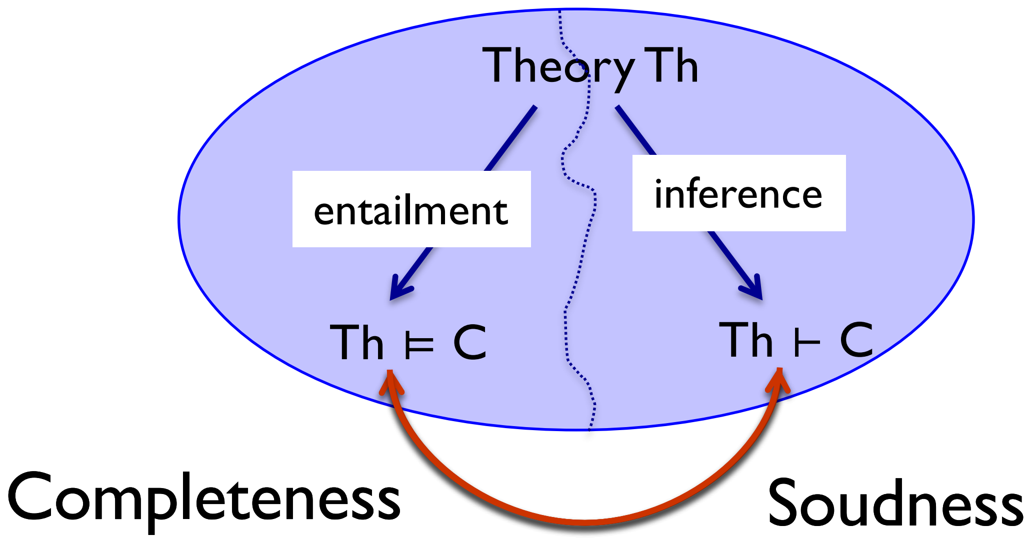
» interpretation of sentences is I = <D, i > where D is a universe of discourse and i maps:

- constants to objects in D
- functions to functions over D
- predicates to tuples over D

» an interpretation and variable assignment satisfy a sentence if given the assignment the sentence is interpreted to be true. A sentence is satisfied if there is an interpretation and variable assignment that satisfy it.

Computational Logic

Predicate Logic helps modeling human reasoning



Make computation logical

Expresses relations between things using logic. Programs describe *what* to compute instead of *how* to compute

Make logic computational

Develop practical algorithms for a subset of logic that is computationally tractable.

Three forms of logic-based reasoning

Deduction

Reasoning from the general to reach the particular:
what follow necessarily from given premises.

Induction

Reasoning from the specifics to reach the general:
process of deriving reliable *generalisations* from observations.

Abduction

Reasoning from observations to *explanations:*
process of using given general rules to establish *causal* relationships between existing knowledge and observations.

Three forms of logic-based reasoning

Deduction

Rule	*All beans in this bag are white*
Case	*These beans are from this bag*
Results	*These beans are white*

Induction

Case	*These beans are from this bag*
Results	*These beans are white*
Rule	*All beans in this bag are white*

Abduction

Rule	*All beans in this bag are white*
Results	*These beans are white*
Case	*These beans are from this bag*

Clausal Representation

- Formulae in special form
  - Theory: set (conjunction) of clauses $\{p \vee \neg q; r; s\}$
  - Clause: disjuction of literals $p \vee \neg q$
  - Literal: atomic sentence or its negation $p \quad \neg q$

- Every formula can be converted into a clausal theory

$(p \vee q) \rightarrow \neg p$		CNF
$\neg(p \vee q) \vee \neg p$	eliminate $\rightarrow$	
$(\neg p \wedge \neg q) \vee \neg p$	push the $\neg$ inwards	
$(\neg p \vee \neg p) \wedge (\neg q \vee \neg p)$	distribute $\vee$ over $\wedge$	
$\neg p \wedge (\neg q \vee \neg p)$	collect terms: $\neg p \vee \neg p$ gives $\neg p$	

What about formulae in Predicate Logic?

Clausal Representation

- Atomic sentences may have terms with variables
  - Theory: set (conjunction) of clauses $\{p(X) \vee \neg r(a,f(b,X))\ ;\ q(X,Y)\}$
    - All variables are understood to be universally quantified

$$\forall X, Y\ [\ (r(a,f(b,X)) \rightarrow p(X))] \wedge \forall X, Y\ q(X,Y)$$

- Substitution $\theta = \{v_1/t_1, v_2/t_2, v_3/t_3, \ldots\}$

  if l is a literal, $l\theta$ is the resulting literal after substitution

  $\theta = \{X/a, Y/g(b,Z)\}$ $p(X,Y)\theta = p(a, g(b,Z))$

- A literal is *ground* if it contains no variables
- A literal l' is *an instance of l,* if for some θ, $l' = l\theta$

Clausal Representation

- Conversion in CNF
  - Skolemisation $\exists X\ p(X) \Rightarrow p(c)$ new constant
    $\forall X\ \exists Y\ p(X,Y) \Rightarrow \forall X\ p(X,f(X))$
  - Remove universal quantifiers

$\forall X(\neg literate(X) \rightarrow (\neg write(X) \wedge \neg \exists Y(book(Y) \wedge read(X,Y))))$

$\forall X(literate(X) \vee (\neg write(X) \wedge \neg \exists Y(book(Y) \wedge read(X,Y))))$ eliminate $\rightarrow$

$\forall X(literate(X) \vee (\neg write(X) \wedge \forall Y(\neg(book(Y) \wedge read(X,Y)))))$ push the $\neg$ inwards

$\forall X(literate(X) \vee (\neg write(X) \wedge \forall Y(\neg book(Y) \vee \neg read(X,Y))))$

$\forall X,Y(literate(X) \vee (\neg write(X) \wedge (\neg book(Y) \vee \neg read(X,Y))))$ remove $\forall$ quantifier

$literate(X) \vee (\neg write(X) \wedge (\neg book(Y) \vee \neg read(X,Y)))$

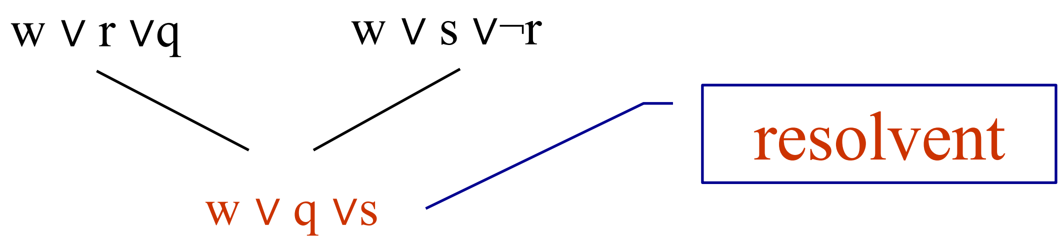
$(literate(X) \vee \neg write(X)) \wedge (literate(X) \vee \neg book(Y) \vee \neg read(X,Y)))$ distribute $\vee$

$\neg write(X)) \vee literate(X)$

$\neg book(Y) \vee \neg read(X,Y) \vee literate(X)$

Propositional resolution

- Given two clauses of the form $p \vee C_1$ and $\neg p \vee C_2$, then the clause $C_1 \vee C_2$ is the inferred clause, called resolvent.



- Resolution is refutation complete

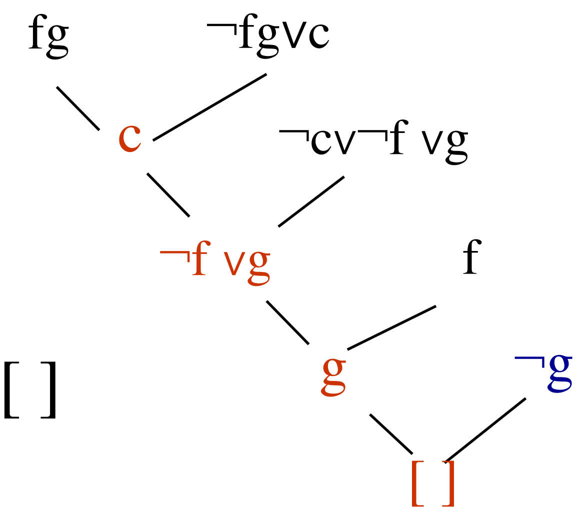
$$Th \models [] \quad \text{iff} \quad Th \vdash []$$

KB

fg
fg → c
c ∧ m → b
c ∧ f → g
f

$\models$ g

$$KB \cup \{\neg g\} \models [\ ]$$



Predicate logic resolution

Main idea: a literal (with variables) stands for all of its instances; so we can allow to infer all such instances in principle.

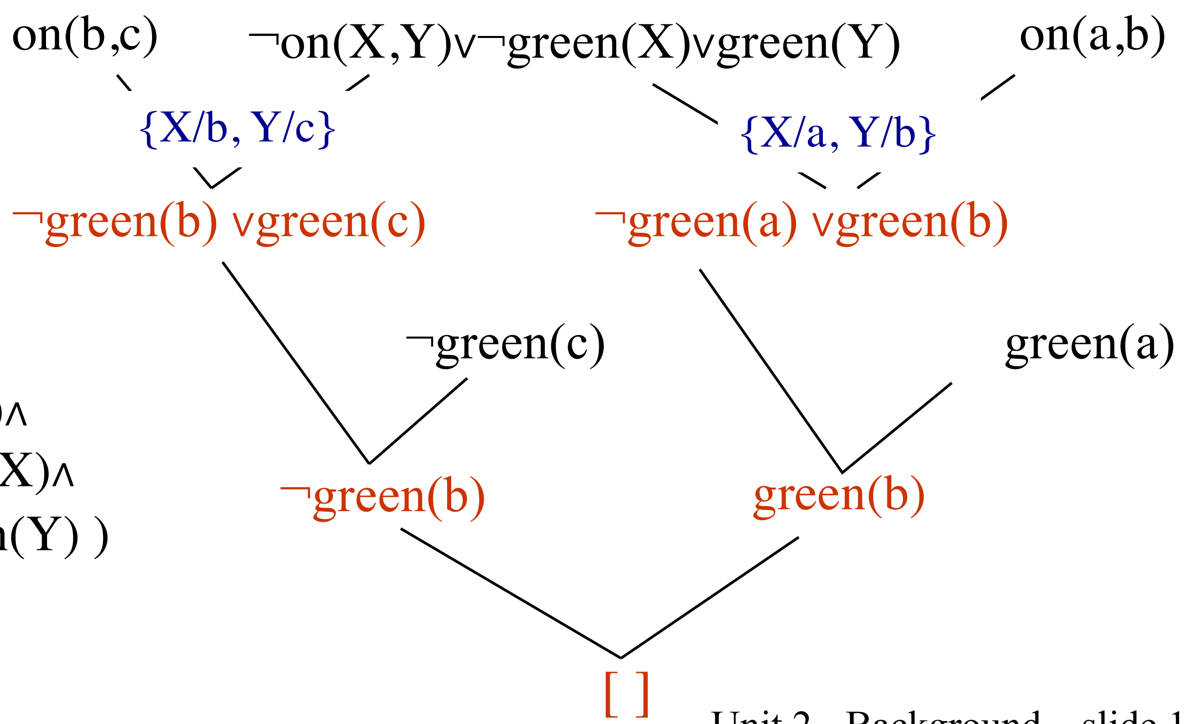
- Given two clauses of the form $\varphi_1 \vee C_1$ and $\neg\varphi_2 \vee C_2$,
  - rename variables so that they are distinct in the two clauses φ_1 and $\neg\varphi_2$
  - for any θ such that $\varphi_1\theta = \varphi_2\theta$, then infer $(C_1 \vee C_2)\theta$
    - φ_1 unifies with φ_2 and θ is the unifier of the two literals

Example

KB

on(a,b)
on(b,c)
green(a)
¬green(c)

$\models \exists X \exists Y\, (\, on(X,Y) \wedge green(X) \wedge \neg green(Y)\,)$



Predicate logic resolution

Answer to the query may return unification values as well

KB

plus(0,X,X)
plus(X,Y,Z) $\rightarrow$ plus(succ(X), Y, succ(Z))

✓ $\models \exists U$ plus(2, 3, U) U = 5

$\neg$plus(X,Y, Z)$\vee$plus(succ(X),Y, succ(Z)) $\neg$plus(2, 3, U)

{X/1, Y/3, U/succ(V), Z/V}

$\neg$plus(1,3,V)

{X/0, Y/3, V/succ(W), Z/W}

$\neg$plus(0,3,W) plus(0, X, X)

{X/3, W/3}

[ ]

Herbrand Theorem

Some predicate cases can be handled by converting them to a propositional form

Given a set Th of clauses

- Herbrand Domain of Th is the set of all ground terms formed using only the constants and function symbols that appear in Th.
- Herbrand Base of Th is the set of all ground atoms that can be formed using the predicate symbols in Th and ground terms in the Herbrand Domain.
- Grounding of Th is the set of all $c\theta$ ground clauses such that $c \in Th$ and the unifier θ replaces variables in c by terms in the Herbrand Domain.

Theorem

A clausal theory Th is satisfiable iff the grounding of Th is satisfiable

Note: Grounding of Th has no variable so it is essentially propositional.

Horn Clauses

Particular types of clauses with at most one positive literal.

definite clauses exactly one positive literals $\neg b_1 \vee \neg b_2 \vee \ldots \vee \neg b_n \vee h$

denials no positive literals $\neg b_1 \vee \neg b_2 \vee \ldots \vee \neg b_n$

Definite clauses can be represented as rules/facts. Denials as constraints:

$\neg b_1 \vee \neg b_2 \vee \ldots \vee \neg b_n \vee h$	$h \leftarrow b_1, b_2, \ldots, b_n$	(rule)
h	h	(fact)
$\neg b_1 \vee \neg b_2 \vee \ldots \vee \neg b_n$	$\leftarrow b_1, b_2, \ldots, b_n$	(constraint)

Herbrand Interpretation of a set Th of definite clauses is a set of ground atoms over the constant, function and predicate symbols occurring in Th.

Herbrand Model: an Herbrand interpretation I is a model of a set Th of clauses iff for all clauses $\neg b_1 \vee \neg b_2 \vee \ldots \vee \neg b_n \vee h_1 \vee \ldots \vee h_m$ in Th and ground substitutions θ,

$$\text{if } \{b_1\theta, b_2\theta, \ldots, b_n\theta\} \in I \text{ then } \{h_1\theta, \ldots, h_m\theta\} \cap I \neq \emptyset$$

Least Herbrand Model

Some set of clauses may have multiple Herbrand models, and some model may be a subset of another model.

human(X) ← male(X)
human(X) ← female(X)
female(X) ∨ male(X) ← human(X)
human(john)

- *What is the Herbrand Domain?*
- *Example of Herbrand model?*
- *Example of Herbrand interpretation that is not a model?*
- *How many Herbrand models?*

An Herbrand model is a Minimal Herbrand model if and only if none of its subsets is an Herbrand model.

Any satisfiable set Th of definite clauses has a UNIQUE minimal Herbrand model, called the *least Herbrand model*.

SLD derivation

SLD inference rule

$\leftarrow \phi_1, \ldots \phi_n \qquad\qquad \phi_1' \leftarrow \beta_1, \ldots, \beta_n$

$\leftarrow \beta_1\theta, \ldots, \beta_n\theta, \phi_2\theta, \ldots, \phi_n\theta$

where θ is the mgu(ϕ_1, ϕ_1')
ϕ_i and β_j are atoms

SLD derivation

Given a denial (goal) G_0 and a set Th of definite clauses, an SLD-derivation of G_0 from Th is a (possibly infinite) sequence of denials

$$G_0 \overset{C_0}{\Rightarrow} G_1 \quad \cdots \quad G_{n-1} \overset{C_{n-1}}{\Rightarrow} G_n$$

where G_{i+1} is derived directly from G_i and a clause C_i with variables appropriately renamed.

The composition $\theta = \theta_1\theta_2 \cdots \theta_n$ of mgus, defined in each step, gives the substitution computed by the whole derivation.

Example of SLD derivation

KB

$proud(X) \leftarrow parent(X,Y), newborn(Y)$
$parent(X,Y) \leftarrow father(X,Y)$
$parent(X,Y) \leftarrow mother(X,Y)$
$father(adam, mary).$
$newborn(mary).$

$\models \exists Z.\ proud(Z)$

G_0 $\leftarrow proud(Z)$ — $proud(X_0) \leftarrow parent(X_0,Y_0), newborn(Y_0)$ — C1

$\{X_0/Z\}$

G_1 $\leftarrow parent(Z,Y_0), newborn(Y_0)$ — $parent(X_1,Y_1) \leftarrow father(X_1,Y_1)$ — C2

$\{X_1/Z, Y_1/Y_0\}$

G_2 $\leftarrow father(Z,Y_0), newborn(Y_0)$ — $father(adam, mary)$ — C3

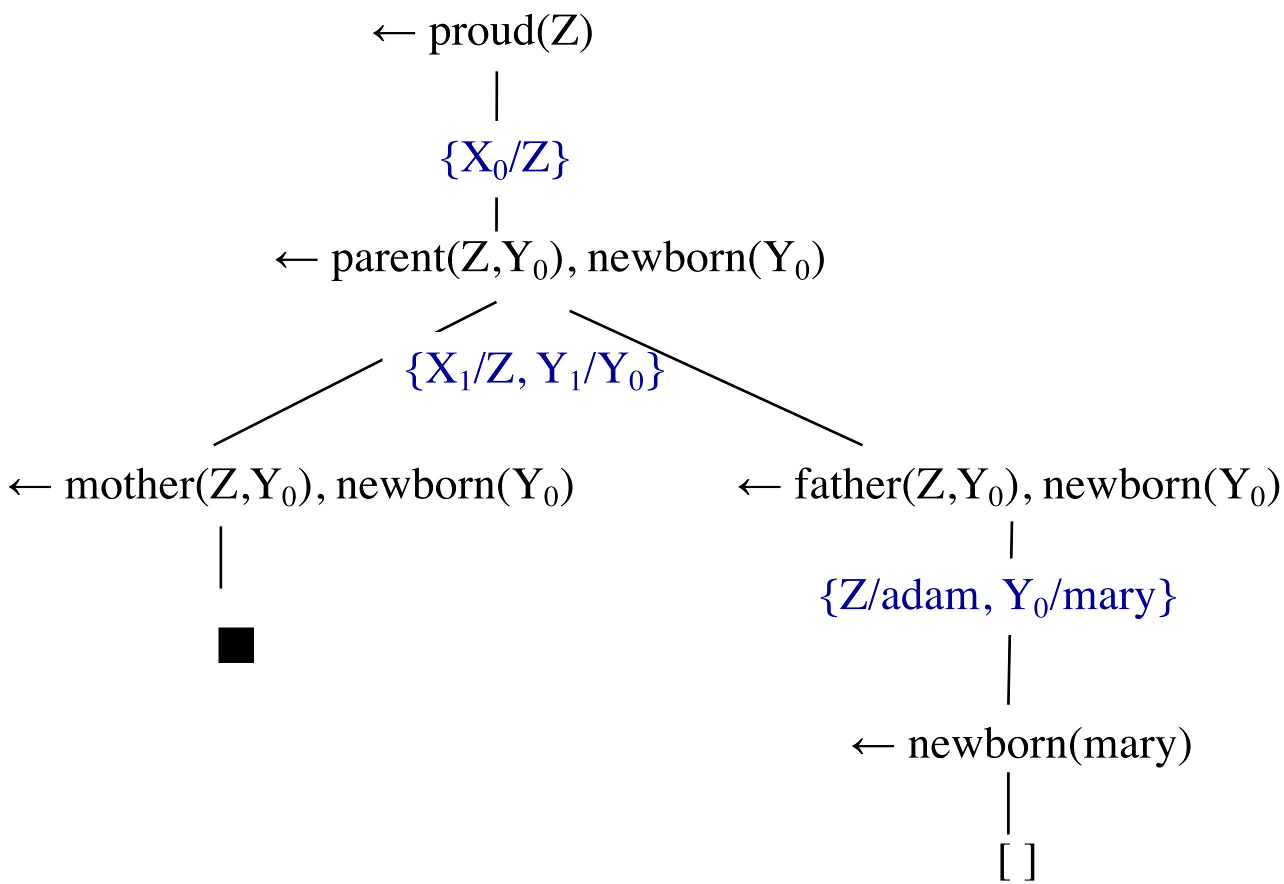
$\{Z/adam, Y_0/mary\}$

G_3 $\leftarrow newborn(mary)$ — $newborn(mary)$ — C4

[ ]

SLD Trees

When selected sub-goal can unify with more than one clause multiple SLD derivations can be computed:



Normal Clausal Logic

Extends Horn Clauses by permitting atoms in the body of rules or of denials to be prefixed with a special operator *not* (read as "fail").

Normal clauses	$h \leftarrow b_1, \ldots, b_n, \textit{not}\ b_{n+1}, \ldots, \textit{not}\ b_m$
Normal denials	$\leftarrow b_1, b_2, \ldots, b_n, \textit{not}\ b_{n+1}, \ldots, \textit{not}\ b_m$

- *not* operator is the \+ used in Prolog.
- computational meaning of *not* p
  - *not* p succeeds if and only if p fails finitely
  - *not* p fails if and only if p succeeds

- fundamental constraint:

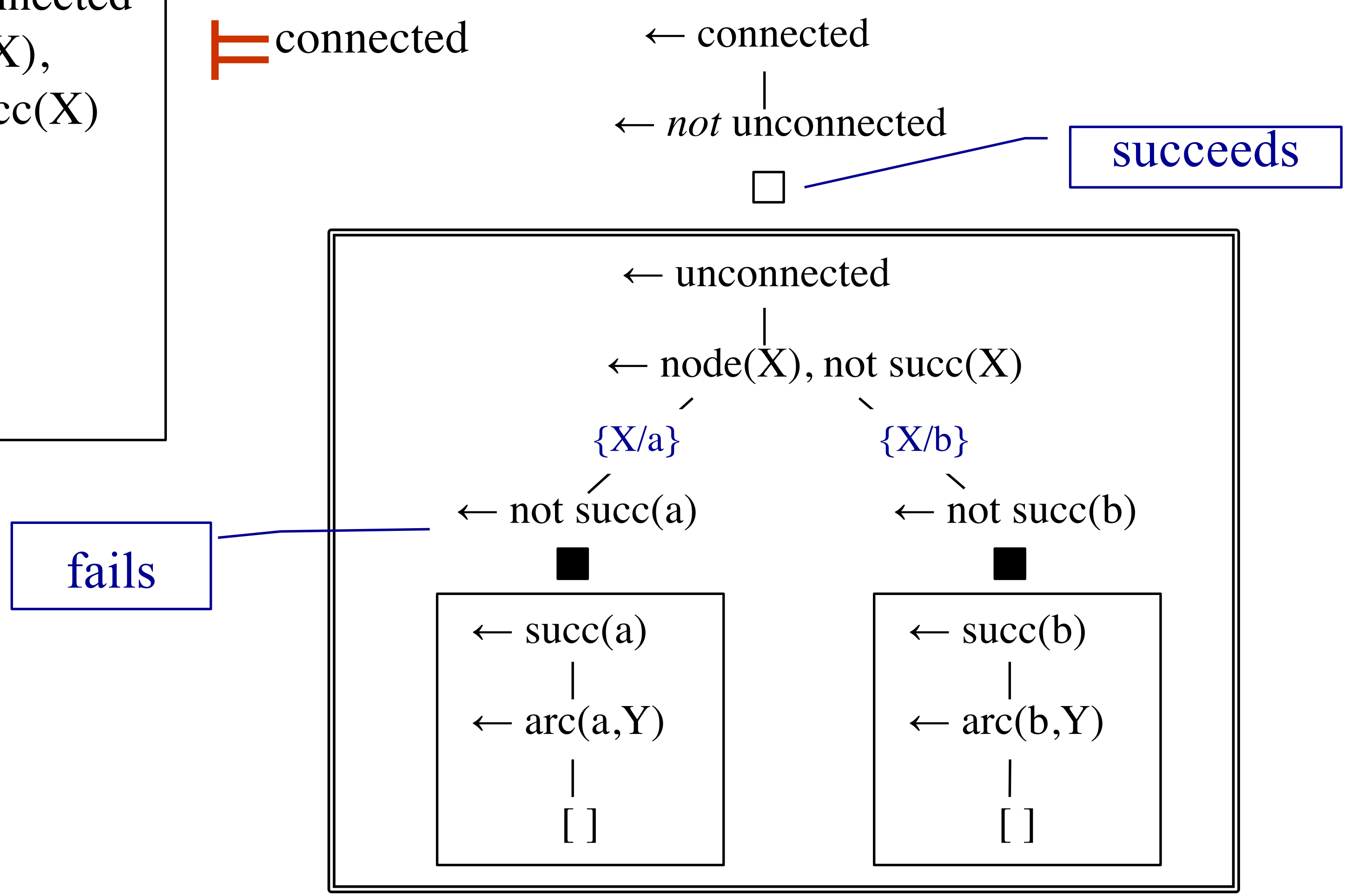
when executing *not* p, p must be ground

SLDNF derivation

We omit a formal definition of an SLDNF derivation.

KB

connected ← *not* unconnected
unconnected ← node(X),
not succ(X)
succ(X) ← arc(X,Y)
node(a)
node(b)
arc(a, b)
arc(b, c)



SLDNF derivation

We omit a formal definition of an SLDNF derivation.

KB

connected ← *not* unconnected
unconnected ← node(X),
not arc(X,Y)
succ(X) ← arc(X,Y)
node(a)
node(b)
arc(a, b)
arc(b, c)

$\not\models$ connected

← connected
|
← *not* unconnected
floundered

← unconnected
|
← node(X), not arc(X,Y)
{X/a} {X/b}
← not arc(a,Y) ← not arc(b,Y)
| |
floundered floundered

Any floundered branch in a tree containing no success branch (refutation) must flounder the node in the parent tree

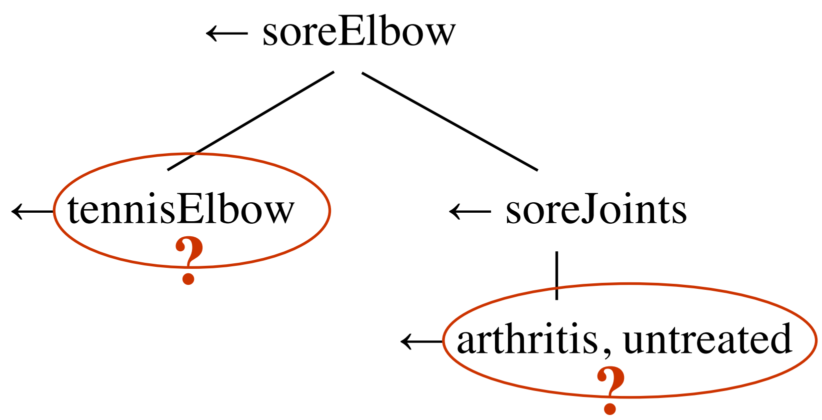
Abduction

So far reasoning has been primarily *deductive*. What about if our knowledge base is incomplete?

KB

soreElbow ← tennisElbow
tennisPlayer ← tennisElbow
soreElbow ← soreJoints
soreJoint ← arthritis, untreated

$\models$ soreElbow



Deductive inference would fail, due to lack of information.
We could assume (as possible hypothesis) what is not known.

Different type of question: "What would explain soreElbow?"

Multiple equally good explanations:

Δ_1 = {tennisElbow}

Δ_2 = {arthritis, untreated}

Abductive reasoning computes explanations of observations with respect to given KB

Abduction

An abductive model of a problem domain is:

$$\langle KB, Ab, IC \rangle \quad \begin{cases} KB = \text{set of normal clauses} \\ Ab = \text{set of ground undefined literals} \\ IC = \text{set of normal denials} \end{cases}$$

Given an abductive model, an abductive solution (called explanation) of a given observation O is a set Δ of ground literals such that:

- $\Delta \sqsubseteq Ab$ belong to a predefined language of abducibles
- $KB \cup \Delta \models O$ provide missing information needed to prove observation
- $KB \cup \Delta \not\models \perp$ is consistent with knowledge base
- $KB \cup \Delta \models IC$ it entails the constraints

Abduction

Consider the following example:

KB	Ab	O
$wobblyWheel \leftarrow brokenSpokes$	brokenSpokes	wobblyWheel
$wobblyWheel \leftarrow flatTyre$	puncturedTube	
$flatTyre \leftarrow leakyValve$	leakyValve	
$flatTyre \leftarrow puncturedTube$		

$$\left.\begin{array}{l} \Delta_1 = \{brokenSpokes\} \\ \Delta_2 = \{leakyValve\} \\ \Delta_3 = \{puncturedTube\} \end{array}\right\} \text{Alternative explanations}$$

Abduction

Consider the following example:

KB

wobblyWheel ← brokenSpokes
wobblyWheel ← flatTyre
flatTyre ← leakyValve
flatTyre ← puncturedTube
smoothRide.
← puncturedTube, smoothRide

Ab

brokenSpokes
puncturedTube
leakyValve

O

wobblyWheel

$$\Delta_1 = \{brokenSpokes\}$$

$$\Delta_2 = \{leakyValve\}$$

$$\cancel{\Delta_3 = \{puncturedTube\}}$$

Constraints may eliminate explanations

Abduction

Consider the following example:

KB

wobblyWheel ← brokenSpokes
wobblyWheel ← flatTyre
flatTyre ← leakyValve
flatTyre ← puncturedTube.
← *not* puncturedTube, leakyValve

Ab

brokenSpokes
puncturedTube
leakyValve

O

wobblyWheel

$$\left.\begin{array}{l}\Delta_1 = \{\text{brokenSpokes}\} \\ \Delta_2 = \{\text{leakyValve, puncturedTube}\} \\ \Delta_3 = \{\text{puncturedTube}\}\end{array}\right\}$$

Constraints may force abducibles in explanations

Abductive reasoning

KB

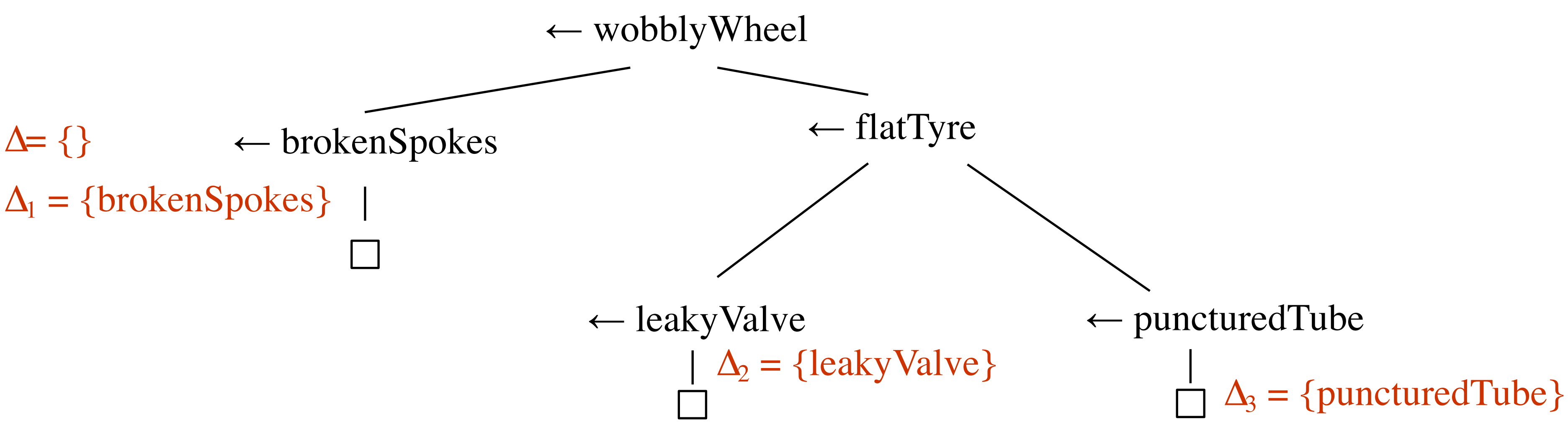
wobblyWheel ← brokenSpokes
wobblyWheel ← flatTyre
flatTyre ← leakyValve
flatTyre ← puncturedTube

Ab

brokenSpokes
puncturedTube
leakyValve

O

wobblyWheel



Abductive reasoning

KB

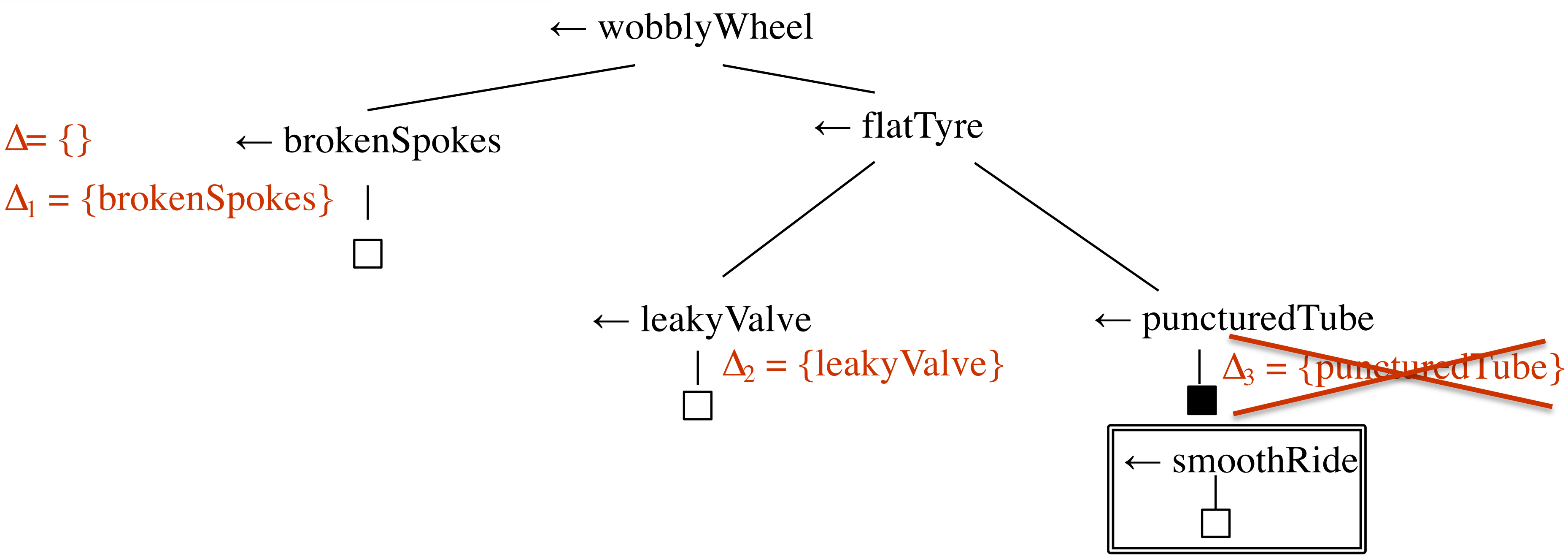
wobblyWheel ← brokenSpokes
wobblyWheel ← flatTyre
flatTyre ← leakyValve
flatTyre ← puncturedTube
smoothRide.
← puncturedTube, smoothRide

Ab

brokenSpokes
puncturedTube
leakyValve

O

wobblyWheel



Abductive reasoning

KB

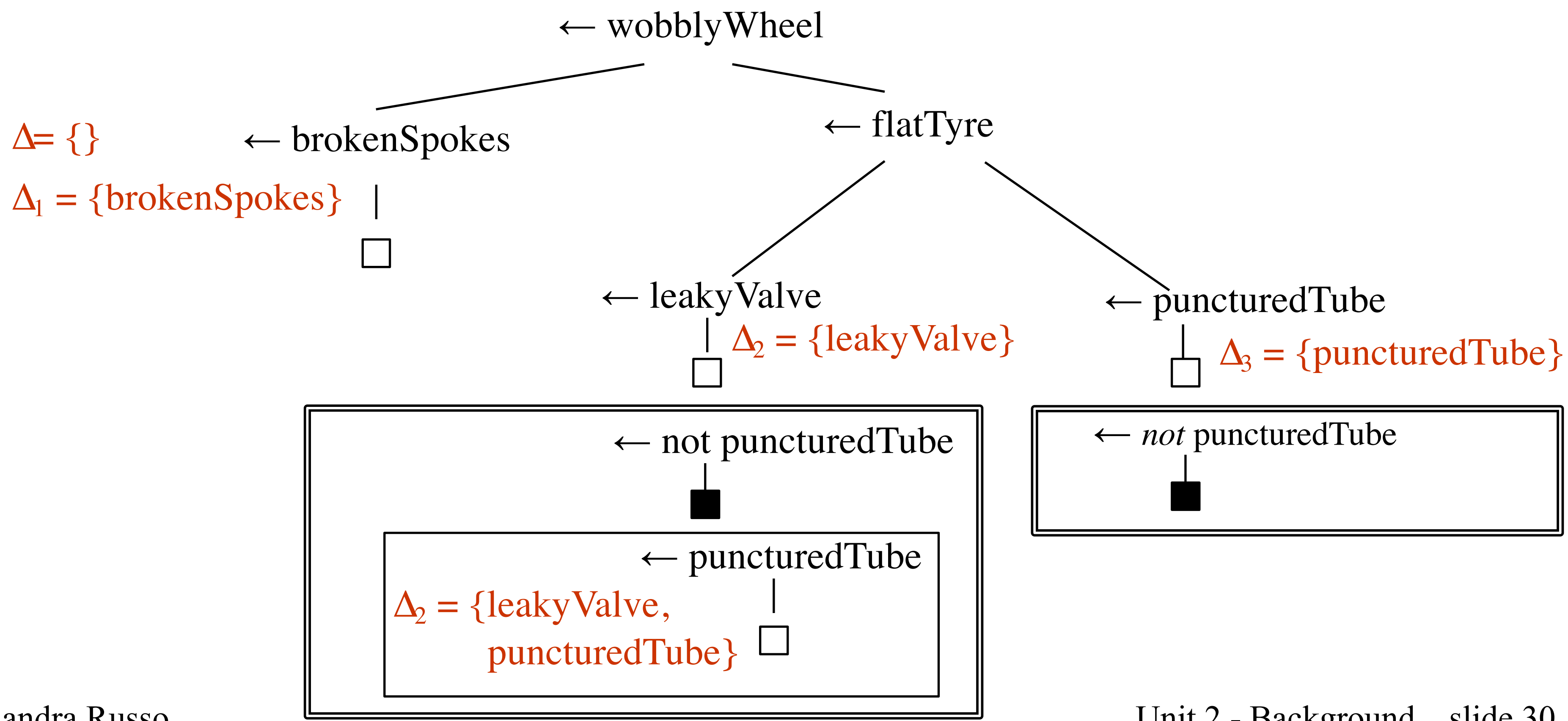
wobblyWheel ← brokenSpokes
wobblyWheel ← flatTyre
flatTyre ← leakyValve
flatTyre ← puncturedTube
←*not* puncturedTube, leakyValve

Ab

brokenSpokes
puncturedTube
leakyValve

O

wobblyWheel



Abductive Proof Procedure

Let <KB, Ab, IC> be an abductive model expressed in normal clausal logic and let O be a ground observation:

Abductive derivation

(G_1, Δ_1)

(G_2, Δ_2)

$\vdots$

$(\emptyset, \Delta_n)$

$G_1 = O$, initially $\Delta = \{\}$

Select a subgoal L from G_i; let $G_i' = G_i - \{L\}$

- $L \notin Ab$ and L is a positive atom

  if $H \leftarrow B$ in KB such that $L = H\theta$

  $G_{i+1} = B\theta \cup G_i'$ and $\Delta_{i+1} = \Delta_i$
- $L \in \Delta_i$ then $G_{i+1} = G_i'$ and $\Delta_{i+1} = \Delta_i$
- $L \in Ab$ and $L \notin \Delta_i$ and *not* $L \notin \Delta_i$

then

$(\Delta_{i+1} = \Delta')$

Consistency derivation

$(L, \Delta_i \cup \{L\})$

$\vdots$

$(\emptyset, \Delta')$

Abductive Proof Procedure

Let <KB, Ab, IC> be an abductive model expressed in normal clausal logic and let O be a ground observation:

Consistency derivation

(A, Δ_1)

(F_1, Δ_2)

$\vdots$

(F_n, Δ_2)

F_1 all denials in IC resolved with A

Select a denial $\leftarrow \phi$ in F_1 and a literal L from it.

- $L \notin Ab$, perform SLDNF failure with L as a subgoal
- $L \in \Delta_i$ and consider new constraint $\leftarrow \phi'$ where $\phi' = \phi - \{L\}$
- $L \in Ab$ and *not* $L \in \Delta_i$ then consider remaining denials in F_1
- $L \in Ab$ and $L \notin \Delta_i$ and *not* $L \notin \Delta_{ii}$ then

$(\Delta_{i+1} = \Delta')$

Abductive derivation

$(not\ L, \Delta_i)$

$\vdots$

$([\,], \Delta')$

Example: Abduction for Normal Clauses

KB

p(X)← *not* q(X)
q(X)← b(X)
←*not* p(X), p(X)
←*not* q(X), q(X)
←*not* b(X), b(X)

Ab

b(a), *not* b(a)
not p(a)
not q(a)

O

p(a)

Abductive solution

Δ = {not q(a), not b(a)}

abductive phase

← p(a)

← *not* q(a)

□

consistency phase

← q(a) Δ_1 = {not q(a)}

← b(a)

■

abductive phase

← *not* b(a) Δ_1 = {not q(a)}

□

consistency phase

← b(a) Δ_2 = {not q(a), not b(a)}

■

Summary

- Summarised propositional and predicate logic.
- Two of types of formal reasoning:
  deduction and abduction.
- Resolution: one of the main deductive proof procedures used in computational logic.
- Focused on Horn clauses and SLD resolution.
- Illustrated SLDNF for normal clauses
- Abductive inference; role of constraints in an abductive proof procedure