Introduction to Artificial Intelligence

Francesca Toni (ft)

Abduction and Argumentation

- Poole and Mackworth – section 5.6

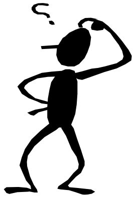
Outline

- Abduction in AI
- From Abduction to Argumentation

I will use logic-programming conventions for variables, terms, predicates

Abduction: example

shoes are wet



shoes are wet if grass is wet

grass is wet if it rained

grass is wet if sprinkler was on

it rained

but what if: *cloudless sky*

and: *if cloudless sky & it rained then false* ?

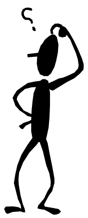
Abduction is non-deterministic, fallacious, non-monotonic

Terminology

From B (e.g. shoes are wet)

B if A (e.g. *shoes are wet if it rained*)

infer A (e.g. it rained)



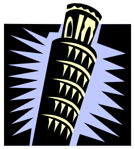
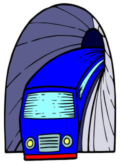
B is an ***observation***

A is an ***assumption/hypothesis/abducible***

that ***explains*** B

A is an ***explanation*** for B

Abduction for AI: many applications

- Planning:
  - observations are goals (e.g. )
  - explanations are plans (e.g.   )
- Diagnosis:
  - observations are symptoms (e.g. toothache)
  - explanations are diseases/faults (e.g. cavity)

- Default reasoning:
  - observations are predictions

    (e.g. Tweety flies / Tweety does not fly)
  - explanations are default rules

    (e.g. birds fly

    penguins do not fly)

Abduction in logic: **Theorist**

Given

- **T** (*theory presentation*), ***FOL theory***
- **H** (*candidate hypotheses*), set of ground ***FOL sentences***
- **O** (*observation*), ***FOL sentence***

E (*explanation*) is such that

1) $T \cup E \models O$

2) $T \cup E$ is **consistent**

(equivalently $T \cup E \not\models$ **false**)

3) $E \subseteq H$

Theorist: example of diagnosis

T: wobbly-wheel $\leftarrow$ broken-spokes $\vee$ flat-tyre
flat-tyre $\leftarrow$ leaky-valve $\vee$ punctured-tube
$\neg$leaky-valve

H: flat-tyre, broken-spokes, leaky-valve, punctured-tube

O: wobbly-wheel

E:

EXPLANATIONS:

- {broken-spokes}, {punctured-tube}, {broken-spokes, punctured-tube} , …

NOT EXPLANATIONS:

- {wobbly-wheel}, {leaky-valve}, {leaky-valve, broken-spokes}, …

Abductive logic programs

- **T** is a ***normal* logic program**
- **H** is a set of ***undefined*** ground **atoms**
- **O** is a (implicitly existentially quantified) conjunction of **literals** (atoms or NAF of atoms)
- **E** is such that

  1) $T \cup E \vdash_{NAF} O$

  2) $E \subseteq H$

Abduction: operational semantics

We have already seen the computation of a goal (query) as a series of derivation steps

$$
\begin{array}{ll}
 & \leftarrow G \\
\theta_1 & \Big| \\
 & \leftarrow G_1 \\
\theta_2 & \Big| \\
 & \vdots \\
 & \leftarrow G_{n-1} \\
\theta_n & \Big| \\
 & \Box
\end{array}
$$

with two kinds of derivation steps, depending on whether the literal selected in the current goal is

a) positive (resolve with a clause…)
b) negative (subcomputation must fail)

Abduction: operational semantics (cntd)

Now:

a) Select a positive literal B
   i. if B not in H: as before
   ii. if B in H and not yet in E: “abduce” it (add it to E)
   iii. if B in H and already in E: throw it away (resolution with E)

b) Select a negative literal *not B*: subcomputation - all ways of computing B from T $\cup$ E must fail finitely, possibly adding to E

Note:

- the computation starts with an empty E
- in the subcomputation, atoms not to be abduced are remembered
- the computed answer is θ + the final E (computed explanation)

Example

T: p ← a, **not** r

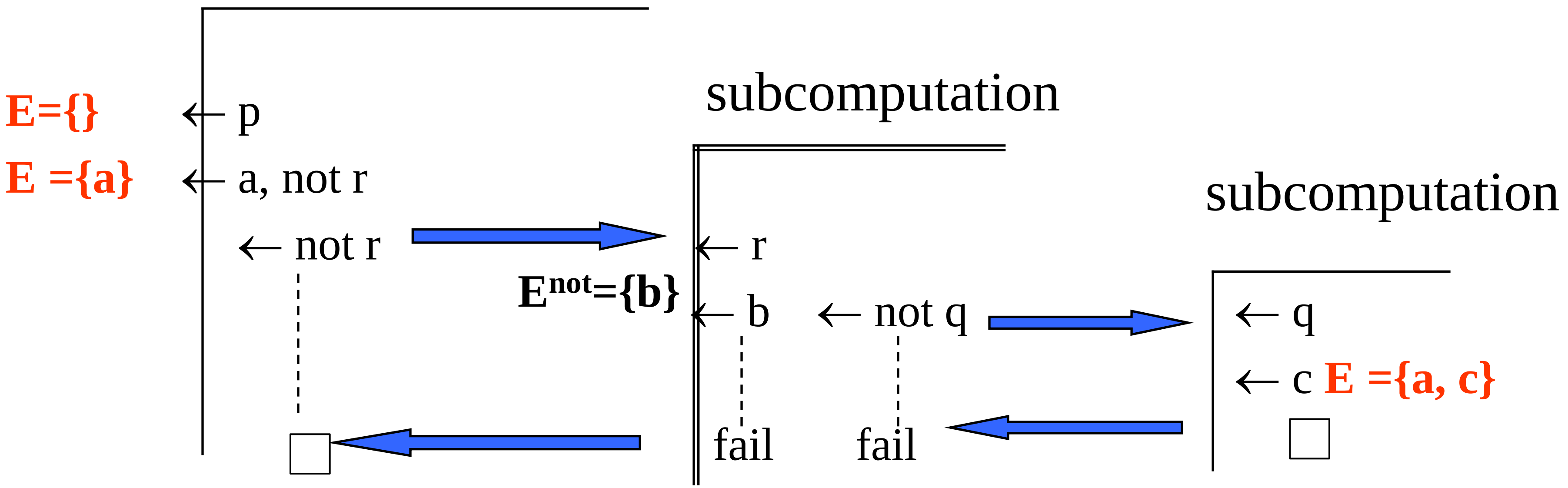
r ← b

r ← **not** q

q ← c

H={a,b,c}

O: p



computed explanation is {a,c}

Another Example

T: p ← a, **not** r, b

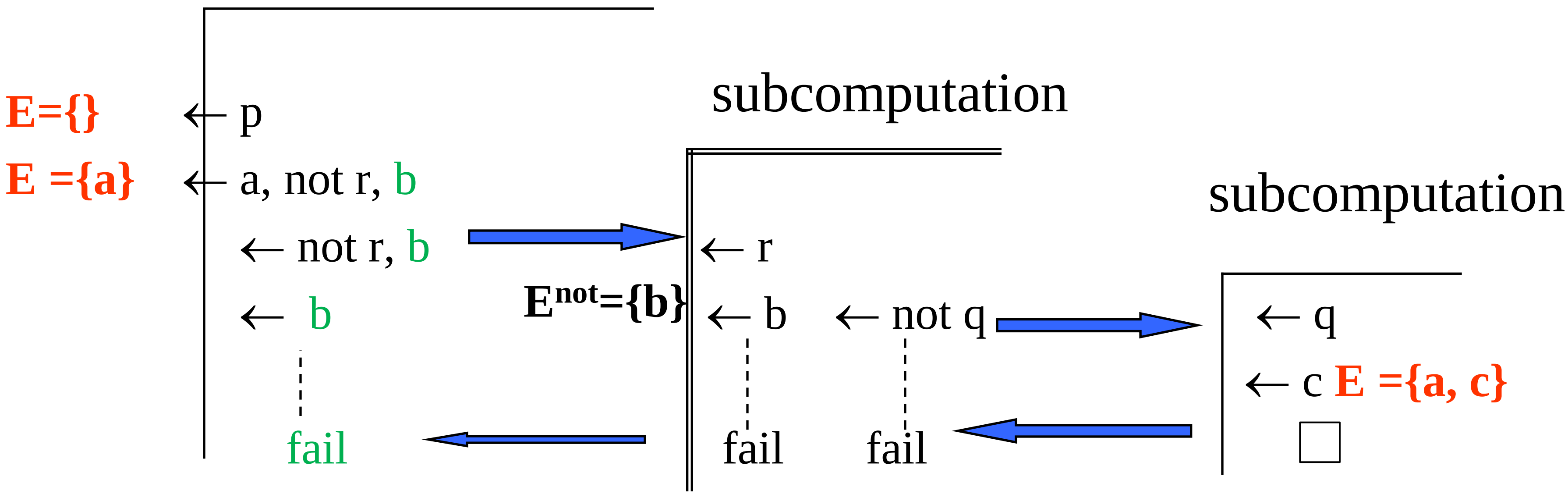
r ← b

r ← **not** q

q ← c

H={a,b,c}

O: p



no computed explanation

From (A)LP to argumentation

- Arguments are deductions from abducibles and negation as failure literals (the *support*) to literals (the *claims*)
- Sub-computations provide counter-arguments and defending arguments
- Derivations are debates (successfully supporting and defending the initial goal/query)

Example: From ALP to argumentation

T: p ← a, **not** r

r ← b

r ← **not** q

q ← c

H={a,b,c}

O: p

Argument:
{not r, a} ⊢ p

Counter-arguments:
{b} ⊢ r, {not q} ⊢ r

Defending-arguments:
{not b} ⊢ not b, {c} ⊢ q

E={} ← p

E ={a} ← a, not r

← not r

$\mathbf{E^{not}=\{b\}}$

← r

← b ← not q

fail fail

← q

← c **E ={a, c}**

$\square$

$\square$

computed explanation is {a,c}

Example: From ALP to argumentation (cntd)

$\longrightarrow$ stands for “attacks” (binary relationship)

$$\begin{tikzcd}
 & \mathbf{\{not\ r,\ a\} \vdash p} & \\
\mathbf{\{b\} \vdash r} \arrow[ur] & & \mathbf{\{not\ q\} \vdash r} \arrow[ul] \\
\mathbf{\{not\ b\} \vdash not\ b} \arrow[u] & & \mathbf{\{c\} \vdash q} \arrow[u]
\end{tikzcd}$$

p succeeds as it is supported by an argument
that can be defended against all counter-arguments:

- counter-arguments attack the argument for p,
- defending arguments attack all counter-arguments
  and cannot be counter-attacked

Example: From LP to argumentation

P: $p \leftarrow q$, **not** r

$r \leftarrow b$

$r \leftarrow$ **not** q

$q \leftarrow$

Goal: p

Argument:

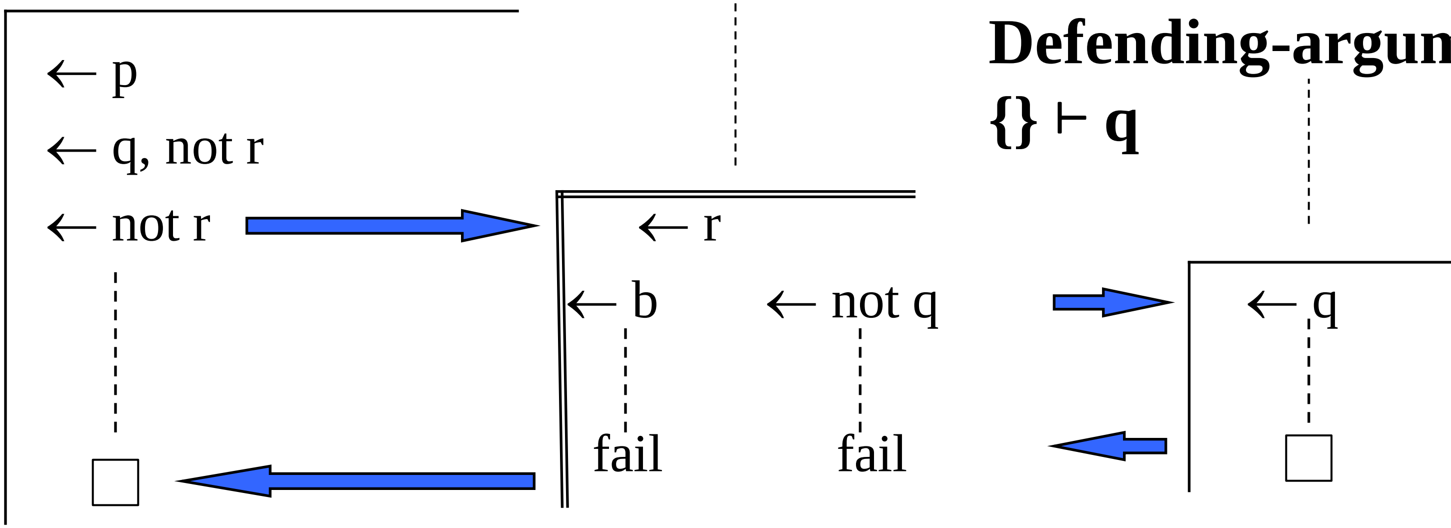
{not r} ⊢ p

Counter-argument:

{not q} ⊢ r

Defending-argument:

{} ⊢ q



computed answer: {}

Example: From LP to argumentation (cntd)

$\longrightarrow$ stands for "attacks" (binary relationship)

$$
\begin{array}{c}
\{\textbf{not } r\} \vdash p \\
\uparrow \\
\{\textbf{not } q\} \vdash r \\
\uparrow \\
\{\} \vdash q
\end{array}
$$

p succeeds as it is supported by an argument
that can be defended against all counter-arguments:

- counter-argument attacks the argument for p,
- defending argument attacks the counter-argument and cannot be counter-attacked

Argumentation semantics (1)

- Operational semantics of normal logic programming/ abduction:
  - A set of arguments A is called *admissible* if it has the *last word* against counter-arguments:
    1. no argument in A attacks any argument in A (A is *conflict-free*)
    2. for every argument b attacking an argument in A, there is an argument in A attacking b
  - If a goal succeeds then there is an admissible set of arguments A including one argument per literal in the goal (the converse does not hold, e.g. see the Nixon diamond again)

Argumentation semantics (2)

- Answer set programming for normal logic programming:
  - A set of arguments A is called *stable* if it attacks all arguments it does not contain:
    1. no argument in A attacks any argument in A (A is *conflict-free*)
    2. for every argument **b** *not* in A there is an argument **a** in A such that **a** attacks **b**
  - given a set of atoms S (from the Herbrand Base of some given P), let A_S be the set of all arguments supported by (subsets of) $\{not\ x \mid x \notin S\}$
  - **S is an answer set iff A_S is stable**

Other uses of argumentation semantics

- Analyse on-line debates (e.g. see www.quaestio-it.com)
- Support design in engineering
- Support decision-making
- …

Summary

- Abduction in AI and Logic (Theorist)
- Abduction in LP (ALP)
- From (A)LP to argumentation