On Preferring and Inspecting Abductive Models

Luís Moniz Pereira[1] Pierangelo Dell'Acqua[2] Gonçalo Lopes[1]

[1] CENTRIA - Centro de Inteligência Artificial
Universidade Nova de Lisboa, Portugal

[2] ITN - Department of Science and Technology
Linköping University, Sweden

IMDEA-Software, Madrid, 25 June 2009

Outline

- Abductive framework
- Declarative semantics
- Pragmatics
- Procedural semantics
- A posteriori preferences
- Medical diagnosis
- Moral decision making
- Implementation
- Conclusions

Declarative Abduction and Preferences Framework

- Abduction is a powerful mechanism to account for defeasible reasoning and incomplete knowledge
- Abductive Logic Programs allow for incompletely defined literals
  - each abductive literal (abducible) can be assumed either true or false in a 2-valued semantics
  - Abductive Logic Programs typically have several models that are derived from the abduced literals
- *Preferences* in Logic Programming have mostly been focused on:
  - preferences among rules of a theory
  - preferences over theory literals

Enabling Pre and Post Preferences

- Our approach is based on handling preference relations between abducibles
- Preferences over abducibles are enacted both a priori or a posteriori wrt. the model generation:
  - *a priori*: to enact preferences during the computation of the models of a theory
  - *a posteriori*: to enact preferences on the computed models of a theory
- Abducibles can be employed as naming devices (cf. Poole) of, say rules, on which preferences are then enacted

Basic Abductive Framework

- Let $\mathcal{L}$ be a first order propositional language defined as follows
- A **literal** can be either an atom A or its default negation *not* A
- A **rule** takes the form:

$$A \leftarrow L_1, \ldots, L_t$$

  where A is an atom and $L_1, \ldots, L_t$ $(t \geq 0)$ are literals
- An **integrity constraint** is a rule whose head is $\bot$:

$$\bot \leftarrow L_1, \ldots, L_t$$

- Each program is associated with a set of abducibles:
  literals which do not occur in any rule head
- To test whether a certain abducible has been abduced, we exploit the reserved abducible

$$abduced(a)$$

  for every abducible $a \neq abduced(.)$
- $abduced(a)$ acts as a constraint that is satisfied in the solution if the abducible a is indeed assumed
- It can be construed as meta-abduction in the form of abducing to check (or passively verify) that a certain abduction is adopted
- $abduced(a)$ permits to check for side-effects of abductions made

Example

Let P be:

$$p \leftarrow abduced(a), a$$
$$q \leftarrow abduced(b)$$

with set of abducibles $\mathcal{A}_P = \{a, b, abduced(a), abduced(b)\}$

P has four intended models:

- $M_1 = \{\}$
- $M_2 = \{p, a, abduced(a)\}$
- $M_3 = \{q, b, abduced(b)\}$
- $M_4 = \{p, q, a, b, abduced(a), abduced(b)\}$

* The set $\{q, abduced(b)\}$ is not an intended model since the assumption of $abduced(b)$ requires the assumption of b

Hypotheses Generation

- Abducibles extend a theory and can be used to provide alternative explanations for a given query
- Generating all alternative explanations for a query is a central problem in abduction because of the combinatorial explosion
- So, generate only those explanations which are preferred and relevant for the query

Enabling the Assumption of Abducibles

- The notion of expectation is employed to express preconditions for enabling the assumption of abducibles
- A considered abducible can be assumed only if there is an expectation for it and, furthermore, there is no expectation to the contrary:

$$\begin{aligned} expect(a) &\leftarrow L_1, \ldots, L_t \\ expect\_not(a) &\leftarrow L_1, \ldots, L_t \end{aligned}$$

  for every abducible $a \neq abduced(.)$
- That is, assumed abducibles are those whose consideration is expected and not defeated

Example Let P be:

$$p \leftarrow a$$
$$q \leftarrow b$$
$$expect(a)$$
$$expect(b)$$
$$expect\_not(a) \leftarrow q$$

with $\mathcal{A}_P = \{a, b, abduced(a), abduced(b)\}$

P has three intended models, for whenever Abd is abduced $abduced(Abd)$ is intended as well:

- $M_1 = \{expect(a), expect(b)\}$
- $M_2 = \{p, a, abduced(a), expect(a), expect(b)\}$
- $M_3 = \{q, b, abduced(b), expect(a), expect(b)\}$

* It is not possible to assume both a and b because the assumption of b makes q true which in turn makes $expect\_not(a)$ true preventing a to be assumed; i.e. the consideration of a is defeated

Enforced Abduction

- Needed to express that the assumption of an abducible enforces the assumption of another abducible
- *Active abduction behavior*

  Let $a, b \neq abduced(.)$ be abducibles

  The reserved atom $a \prec b$ states that the assumption of b enforces the assumption of a

  that is, if b is assumed, then $a \prec b$ forces a to be assumed provided that a can be considered

Example Let P be:

$$p \leftarrow a$$
$$a \prec b$$
$$b \prec a$$
$$expect(a)$$
$$expect(b)$$

with set of abducibles $\mathcal{A}_P = \{a, b, abduced(a), abduced(b)\}$

P has two intended models:

- $M_1 = \{a \prec b, b \prec a, expect(a), expect(b)\}$
- $M_2 = \{p, a, b, a \prec b, b \prec a, expect(a), expect(b), abduced(a), abduced(b)\}$

* This is due to the active abduction behavior of $a \prec b$ and $b \prec a$ that prevents intended models containing just either a or b

Conditional Abduction

- The assumption of an abducible a can be conditional on the assumption of another abducible b
- *Passive abduction behavior*

  Let $a, b \neq abduced(.)$ be abducibles

  The reserved atom $a \sqsubset b$ states that a can be assumed only if b is (without assuming it for the purpose of having a)
  That is, $a \sqsubset b$ acts as a check passively constraining the assumption of a to the assumption of b

Example Let P be:

$$p \leftarrow a$$
$$q \leftarrow b$$
$$a \sqsubset b$$
$$expect(a)$$
$$expect(b)$$

with set of abducibles $\mathcal{A}_P = \{a, b, abduced(a), abduced(b)\}$

P has three intended models:

- $M = \{a \sqsubset b, expect(a), expect(b)\}$
- $M_2 = \{a \sqsubset b, b, q, expect(a), expect(b), abduced(b)\}$
- $M_3 = \{p, q, a, b, a \sqsubset b, expect(a), expect(b), abduced(a), abduced(b)\}$

* The global $a \sqsubset b \leftarrow cond$ can be avoided altogether by replacing every occurrence of the abducible a with "$cond, a, abduced(b)$"

Cardinality Constrained Abduction

- Allow to constrain the number of assumed abducibles
- *Active abduction behavior*
  The reserved atom $L\ \{l_1, \ldots, l_n\}\ U$ states that at least L and at most U abducibles in $\{l_1, \ldots, l_n\}$ must be assumed; does not assume them, just checks whether they are abduced within such lower and upper bounds
- ∗ The abducibles l_i can also take the form $abduced(.)$

Declarative Semantics

- Given in terms of abductive stable models
- A (2-valued) **interpretation** M is any set of literals such that, for every atom A, precisely one of the literals A or *not* A belongs to M
- Set of **default assumptions**:

$$Default(P, M) = \{not\ A : \text{there exists no rule } A \leftarrow L_1, \ldots, L_t \text{ in } P \text{ such that } M \models L_1, \ldots, L_t\}$$

* Abducibles are false by default since we made the assumption that abducibles are not defined by any rule in P

- An interpretation M is a **stable model** (SM) of P iff:
  - $M \not\models \bot$
  - $M = least(P \cup Default(P, M))$

  where *least* indicates the least model
- Let C be $L\,\{l_1, \ldots, l_n\}\,U$

  Let $W(C, M)$ be the number of abducibles in $\{l_1, \ldots, l_n\}$ satisfied by an interpretation M:

$$W(C, M) = |\,\{l : l \in \{l_1, \ldots, l_n\} \text{ and } M \models l\}\,|$$

- Given a set of abducibles Δ, we write Δ^* to indicate:

$$\Delta^* = \{a \ : \ a \neq abduced(.) \text{ and } a \in \Delta\}$$

Let $\Delta \subseteq \mathcal{A}_P$. M is an **abductive stable model** with hypotheses Δ of P iff:

1. $M \not\models \bot$
2. $M = least(Q \cup Default(Q, M))$, where $Q = P \cup \Delta$
3. $M \models expect(a)$ and $M \not\models expect\_not(a)$, for every $a \in \Delta^*$
4. for every $a \in \Delta^*$, if $M \models a$ then $M \models abduced(a)$
5. for every atom $a \prec b$, if $M \models a \prec b$, $M \models expect(a)$, $M \not\models expect\_not(a)$ and $M \models b$, then $M \models a$
6. for every atom C of the form $L \{l_1, \ldots, l_n\} U$, if $M \models C$ then $L \leq W(C, M) \leq U$
7. for every $a \in \Delta^*$, if $M \models abduced(a)$ then $M \models a$
8. for every atom $a \sqsubset b$, if $M \models a \sqsubset b$ and $M \models a$, then $M \models b$

Example Let P be:

$$p \leftarrow abduced(a)$$
$$expect(a)$$

with set of abducibles $\mathcal{A}_P = \{a, abduced(a)\}$

Then, $M_1 = \{p, a, abduced(a), expect(a)\}$ is an abductive SM with hypotheses $\Delta = \{a, abduced(a)\}$

while $M_2 = \{p, abduced(a), expect(a)\}$ is not since condition (7) of Abductive SM is not fulfilled

- Let G be a goal. Then Δ is an **abductive explanation** for G in P iff:
  - M is an abductive SM with hypotheses Δ of P, and
  - $M \models G$

- Let G be a goal. Then, Δ is a **strict abductive explanation** for G in P iff
  (1) Δ is a minimal set for which:
    - $M \models G$
    - $M \not\models \bot$
    - $M = least(Q \cup Default(Q, M))$, where $Q = P \cup \Delta$
    - $M \models expect(a)$ and $M \not\models expect\_not(a)$, for every $a \in \Delta^*$
    - for every $a \in \Delta^*$, if $M \models a$ then $M \models abduced(a)$
    - for every atom $a \prec b$, if $M \models a \prec b$, $M \models expect(a)$, $M \not\models expect\_not(a)$ and $M \models b$, then $M \models a$
    - for every atom C of the form $L\,\{l_1, \ldots, l_n\}\,U$, if $M \models C$ then $L \leq W(C, M) \leq U$

  (2) for every $a \in \Delta^*$, if $M \models abduced(a)$ then $M \models a$
  (3) for every atom $a \sqsubset b$, if $M \models a \sqsubset b$ and $M \models a$, then $M \models b$

* Note that conditions (2) and (3) are not subject to minimization.
  And in (1) only the first item is new; the remainder simply pertain to the abductive stable model conditions.

Example Let P be:

$$p \leftarrow abduced(a)$$
$$expect(a)$$

with set of abducibles $\mathcal{A}_P = \{a, abduced(a)\}$

Let the goal G be $?{-}p$

It holds that $\Delta = \{a, abduced(a)\}$ is an abductive explanation for G in P
Δ is not strict since it is not a minimal set satisfying condition (1) of strict abductive explanation

The minimal set is $\Delta_2 = \{abduced(a)\}$

Hence, there exists no strict abductive explanation for G in P

Property

- The following result relates abductive explanations and strict abductive explanations

  Property. Let G be a goal and Δ a strict abductive explanation for G in P. Then, Δ is an abductive explanation for G in P.

- It is easy to see that the converse does not hold since abductive explanations are not subject to the minimality requirement

Pragmatics

- Constraining Abduction
- Abducible Sets
- Preferring Abducibles
- Modelling Inspection Points

Constraining Abduction

- In domain problems the assumption of abducibles is often subject to the fulfillment of certain conditions, including other assumptions, which must be satisfied
  This requirement can be expressed by constrained abduction, $a \sqsubset b$
- Consider a scenario where:
  - there is a pub that is open or closed
  - if the light is on in the pub, then it is open or being cleaned
  - if late night, the pub is open if there are people inside
  - the pub being located in an entertainment district, there is noise around if there are people in the pub or a party nearby

  This scenario is described by the following program P with

  $\mathcal{A}_P = \{open, cleaning, party, people, abduced(open), abduced(cleaning), abduced(party), abduced(people)\}$

$$light \leftarrow open, not\ cleaning$$
$$light \leftarrow cleaning, not\ open, not\ abduced(people)$$
$$open \sqsubset people \leftarrow late\_night$$
$$noise \leftarrow party$$
$$noise \leftarrow people$$

- If it is night (but not late night) and one does observe lights in the pub, then one has two equally plausible explanations for it: $\{open\}$ or $\{cleaning\}$

$$light \leftarrow open, not\ cleaning$$
$$light \leftarrow cleaning, not\ open, not\ abduced(people)$$
$$open \sqsubset people \leftarrow late\_night$$
$$noise \leftarrow party$$
$$noise \leftarrow people$$

- Otherwise (it is late night), then one will assume only one minimal explanation for the lights being turned on:
  $\{cleaning\}$
- If instead it is late night and one also hears noise (i.e. $?-light, noise$ is true), then one will have now three abductive explanations:
  $\{open, people\}$, $\{cleaning, party\}$ and $\{open, party, people\}$
  where the latter is not minimal

Abducible Sets

- Often it is desirable to express contextual information constraining the powerset of abducibles
- The behavior of abducibles over different sets is highly context-dependent and should be embedded over rules in the theory
- The problem is analogous to the one addressed by *cardinality* and *weight constraint rules* for the Stable Model semantics implementations

- **Example**

  Claire is deciding what to have for a meal from a limited buffet

  The menu has:

  - appetizers which she doesn't mind skipping, unless she's very hungry
  - three main dishes from which she can select a maximum of two
  - drinks from which she will have a single one

  The situation can be modelled as follows

  $\mathcal{A}_P = \{bread, salad, cheese, fish, meat, veggie, wine, juice, water,$
  $abduced(bread), abduced(salad), abduced(cheese), abduced(fish),$
  $abduced(meat), abduced(veggie), abduced(wine), abduced(juice),$
  $abduced(water)\}$

$$0\ \{bread, salad, cheese\}\ 3 \leftarrow appetizers$$
$$1\ \{fish, meat, veggie\}\ 2 \leftarrow main\_dishes$$
$$1\ \{wine, juice, water\}\ 1 \leftarrow drinks$$
$$2\ \{appetizers, main\_dishes, drinks\}\ 3$$
$$main\_dishes \prec appetizers$$
$$drinks \prec appetizers$$
$$appetizers \leftarrow very\_hungry$$

- In this situation we model appetizers as being the least preferred set from those available for the meal
- This shows how one can condition sets of abducibles based on the generation of literals from other cardinality constraints, along with preferences among such literals

Preferring Abducibles

- To express cardinality preferences between considered abducibles, we employ the construct $L\,\langle l_1, \ldots, l_n \rangle\, U$

$$L\,\langle l_1, \ldots, l_n \rangle\, U \equiv L\,\{abduce(l_1), \ldots, abduce(l_n)\}\, U$$

  for any abducible $l_1, \ldots, l_n$ distinct from $abduced(.)$
- It constrains the number of abducibles assumed (without having to assume them)

- Consider a scenario where:
  - three alternative abducibles a, b and c can explain an observation p
  - a and c are preferred to b

This can be axiomatized by the following program P with abducibles: $\mathcal{A}_P = \{a, b, c, abduced(a), abduced(b), abduced(c)\}$

$$p \leftarrow a \qquad 0\ \langle a, b, c\rangle\ 1$$
$$p \leftarrow b$$
$$p \leftarrow c \qquad expect(a)$$
$$expect(b)$$
$$a \prec b \qquad expect(c)$$
$$c \prec b$$

If the preference relation is a strict partial order, then that must be axiomatized explicitly

- P has three abductive stable models:
  - $M_1 = \{a \prec b, c \prec b, 0\,\langle a, b, c\rangle\,1, expect(a), expect(b), expect(c)\}$
    $\Delta_1 = \{\}$
  - $M_2 = \{a \prec b, c \prec b, 0\,\langle a, b, c\rangle\,1, a, p, expect(a), expect(b), expect(c), abduced(a)\}$
    $\Delta_2 = \{a, abduced(a)\}$
  - $M_3 = \{a \prec b, c \prec b, 0\,\langle a, b, c\rangle\,1, c, p, expect(a), expect(b), expect(c), abduced(c)\}$
    $\Delta_3 = \{c, abduced(c)\}$
- The strict abductive explanations for $?-p$ in P are Δ_2 and Δ_3

  Note that b cannot be part of any abductive explanation

- Consider a situation where Claire:
  - drinks either tea or coffee (but not both)
  - prefers coffee over tea when sleepy
  - doesn't drink coffee when she has high blood pressure

  This situation can be represented by a program P with set of abducibles

  $\mathcal{A}_P = \{tea, coffee, abduced(tea), abduced(coffee)\}$

$$drink \leftarrow tea$$
$$drink \leftarrow coffee$$

$$expect(tea)$$
$$expect(coffee)$$
$$expect\_not(coffee) \leftarrow blood\_pressure\_high$$

$$0\ \langle tea, coffee \rangle\ 1$$
$$coffee \prec tea \leftarrow sleepy$$

- P has two models: one with tea and the other with coffee
- By adding *sleepy*, the enforced abduction comes into play, defeating the abductive stable model where only *tea* is present
- If we add *blood_pressure_high* to P, coffee is no longer expected, and the preference rule no longer defeats the abduction of *tea*

Modeling Inspection Points

- In Abductive Logic Programming we wish to find, and choose, abductive solutions for queries
- For that we may need to know side effects of adoptable abductions
- Indeed, preferences regarding abductive solutions can depend on their side effects
- Normally, not all side effects are of interest!
- Pereira and Pinto [IDT09] argue that this type of reasoning requires a new mechanism: the **inspection points**

- Consider the following example
  $\mathcal{A}_P =$
  $\{drink\_water, drink\_beer, abduced(drink\_water), abduced(drink\_beer)\}$

$$\leftarrow thirsty, not\ drink \qquad \text{(IC)}$$

$$thirsty$$
$$drink \leftarrow drink\_water$$
$$drink \leftarrow drink\_beer$$
$$wet\_glass \leftarrow use\_glass$$
$$use\_glass \leftarrow drink$$
$$drunk \leftarrow drink\_beer$$

- We want to satisfy the IC without getting drunk. How?
  Knowing whether *wet_glass* is true is irrelevant

- **Meta-Abduction** is abducing the intention of checking a posteriori for the actual adoption (or not) of some abducible
- It can be the base mechanism for reifying:
  - *Inspection Points*: side-effects check
  - *Intentions*: future goal scheduling
  - *Deontic Verifiers*: moral imposition/prevention of certain abductions
  - *A posteriori Preferences*: preferring abductive solutions with certain side-effects

Example: Inspection Points for checking side-effects

$$\leftarrow thirsty, not\ drink$$

$$thirsty$$
$$drink \leftarrow drink\_water$$
$$drink \leftarrow drink\_beer$$
$$wet\_glass \leftarrow use\_glass$$
$$use\_glass \leftarrow drink$$
$$drunk \leftarrow drink\_beer$$
$$unsafe\_driving \leftarrow inspect(drunk)$$

unsafe_driving is verified if *drink_beer*
But *drink_beer* is not allowed as a solution to the *unsafe_driving* rule alone!
inspect/1 checks for abducibles used, but does not abduce

Example: Inspection Points for Preferring Abductions

$\bot \leftarrow police, riot, not\ contain$

$contain \leftarrow tear\_gas$
$smoke \leftarrow fire$
$police$

$contain \leftarrow water\_cannon$
$smoke \leftarrow inspect(tear\_gas)$
$riot$

$\mathcal{A}_P = \{tear\_gas, fire, water\_cannon\}$

- *fire* is a more plausible explanation for *smoke*
- To let the explanation for *smoke* be *tear_gas*, there must be a plausible reason imposed by some other likely phenomenon
- This is represented by $inspect(tear\_gas)$ instead of simply *tear_gas*
- $inspect(L)$ where L is abducible can be replaced simply by $abduced(L)$

- Inspection points can occur within the proof tree of another inspection point

$$x \leftarrow a, inspect(y), b, c, not\ d \qquad y \leftarrow inspect(not\ a)$$
$$z \leftarrow d \qquad y \leftarrow b, inspect(not\ z), c$$

$\mathcal{A}_P = \{a, b, c, d\}$
Let the query be $?{-}x$

According to the intended semantics of inspection points, the set

$$\Delta = \{a, b, c\}$$

is an abductive explanation for the query

Transforming Inspection Points

- The transformation Π maps a program P with inspection points into a regular abductive program

  The program $\Pi(P)$ consists of:

  - all the rules obtained by the rules in P by replacing: $inspect(a)$ or $inspect(abduced(a))$ with $abduced(a)$ if a is an abducible, and keeping $inspect(L)$ otherwise
  - $inspect(not\, L)$ with $not\ inspect(L)$
  - for every abducible a, the rule: $expect(a)$

- for every rule $A \leftarrow L_1, \ldots, L_t$ in P, the additional rule:

$$inspect(A) \leftarrow L_1^{'}, \ldots, L_t^{'}$$

where for every $1 \leq i \leq t$

$$L_i^{'} = \begin{cases} abduced(L_i) & \text{if } L_i \text{ is an abducible} \\ inspect(X) & \text{if } L_i \text{ is } inspect(X) \\ inspect(L_i) & \text{otherwise} \end{cases}$$

Example: Let P be the program of the previous example. $\Pi(P)$ is:

$x \leftarrow a, inspect(y), b, c, not\ d$
$inspect(x) \leftarrow abduced(a), inspect(y), abduced(b), abduced(c), abduced(not\ d)$

$y \leftarrow abduced(not\ a)$
$inspect(y) \leftarrow abduced(not\ a)$

$y \leftarrow b, inspect(not\ z), c$
$inspect(y) \leftarrow abduced(b), inspect(not\ z), abduced(c)$

$z \leftarrow d$
$inspect(z) \leftarrow abduced(d)$

$expect(a) \qquad expect(b) \qquad expect(c) \qquad expect(d)$

The abductive stable model of $\Pi(P)$ respecting the inspection points is:

$M = \{x, a, b, c, abduced(a), abduced(b), abduced(c), expect(a), expect(b), expect(c), expect(d)\}$

Procedural Semantics

- Framework
- Program Transformation
- Correctness

Framework

- Let $\mathcal{L}^{\#}$ be a first-order propositional language containing rules
- Let Γ and Σ be sets of rules over $\mathcal{L}^{\#}$. Then (Γ, Σ) is a **restricted program**
  - Γ contains the rules formalizing the application domain
  - Σ formalizes the properties that Γ must satisfy
- Every restricted program is associated with a set of abducibles $\mathcal{A}_{(\Gamma,\Sigma)}$

$*$ Note that the definition of restricted program can be generalized to have Σ be any set of wffs

- Let $\Delta \subseteq \mathcal{A}_{(\Gamma,\Sigma)}$ be a set of abducibles
- M is a **valid stable model** with hypotheses Δ of (Γ, Σ) iff:
  1) $M \not\models \bot$
  2) $M = least(\Gamma^+ \cup Default(\Gamma^+, M))$, where $\Gamma^+ = \Gamma \cup \Delta$
  3) $M \models \Sigma$
- A valid stable model is an abductive stable model (conditions (1) and (2)) satisfying the wffs in Σ (condition (3))

- **Example** Let (Γ, Σ) be the restricted program:

$$\Gamma = \{p \leftarrow a; expect(a)\}$$
$$\Sigma = \{q \leftarrow a\}$$

with $\mathcal{A}_{(\Gamma,\Sigma)} = \{a, abduced(a)\}$

Its unique valid stable model is:

$$M = \{expect(a)\}$$

* The only other model could be $M_2 = \{p, a, abduced(a), expect(a)\}$ but it is not valid, Σ not being satisfied

- **Example** Let (Γ, Σ) be:

$$\Gamma = \{p \leftarrow a; expect(a); expect(b)\}$$
$$\Sigma = \{\bot \leftarrow a, not\ b\}$$

with $\mathcal{A}_{(\Gamma,\Sigma)} = \{a, b, abduced(a), abduced(b)\}$

(Γ, Σ) has two valid stable models:

- $M_1 = \{expect(a), expect(b)\}$
- $M_2 = \{p, a, b, abduced(a), abduced(b), expect(a), expect(b)\}$

- Let G be a goal. Then, Δ is a **valid explanation** for G in (Γ, Σ) iff:
  - M is a valid stable model with hypotheses Δ of (Γ, Σ), and
  - $M \models G$
- Let G be a goal. Then, Δ is a **strict valid explanation** for G in (Γ, Σ) iff:
  1) Δ is a minimal set for which:
     - $M \not\models \bot$
     - $M = least(\Gamma^+ \cup Default(\Gamma^+, M))$, where $\Gamma^+ = \Gamma \cup \Delta$
     - $M \models G$
  2) $M \models \Sigma$
- **Proposition** Let G be a goal and Δ a strict valid explanation for G in (Γ, Σ). Then, Δ is an valid explanation for G in (Γ, Σ)

$*$ It is easy to see that the converse of the proposition does not hold

Program Transformation

- We define a transformation γ mapping programs over $\mathcal{L}$ into restricted programs over $\mathcal{L}^{\#}$
- Let P be a program over $\mathcal{L}$ with set of abducibles $\mathcal{A}_P$. The restricted program $\gamma(P) = (\Gamma, \Sigma)$ over $\mathcal{L}^{\#}$ with set of abducibles $\mathcal{A}_{(\Gamma,\Sigma)} = \mathcal{A}_P$ is defined as follows

Γ consists of:

1. all the rules in P
2. $\bot \leftarrow a, not\ expect(a)$
   $\bot \leftarrow a, expect\_not(a)$
   for every abducible $a \in \mathcal{A}_P^*$
3. $\bot \leftarrow a, not\ abduced(a)$
   for every abducible $a \in \mathcal{A}_P^*$
4. $\bot \leftarrow a \prec b, expect(a), not\ expect\_not(a), b, not\ a$
   for every rule $a \prec b \leftarrow L_1, \ldots, L_t$ in P
5. $\bot \leftarrow L\,\{l_1, \ldots, l_n\}\,U, count([l_1, \ldots, l_n], N), N < L$
   $\bot \leftarrow L\,\{l_1, \ldots, l_n\}\,U, count([l_1, \ldots, l_n], N), N > U$
   for every rule $L\,\{l_1, \ldots, l_n\}\,U \leftarrow L_1, \ldots L_t$ in P

Σ consists of:

6. $\bot \leftarrow abduced(a), not\ a$
   for every abducible $a \in \mathcal{A}_P^*$
7. $\bot \leftarrow a \sqsubset b, a, not\ b$
   for every rule $a \sqsubset b \leftarrow L_1, \ldots, L_t$ in P

* We assume given the atom $count([l_1, \ldots, l_n], m)$ that holds if m is the number of abducibles belonging to $[l_1, \ldots, l_n]$ that are assumed:
  if $C \equiv L\,\{l_1, \ldots, l_n\}\,U$, then
  $M \vDash count([l_1, \ldots, l_n], m)$ iff $W(C, M) = m$
  for any interpretation M

* If P contains inspection points, then apply the transformation γ to $\Pi(P)$

- **Example** Consider the coffee and tea program P

  The transformed program $\gamma(P) = (\Gamma, \Sigma)$ over $\mathcal{L}^{\#}$ with abducibles $\mathcal{A}_{(\Gamma,\Sigma)} = \{tea, coffee, abduced(tea), abduced(coffee)\}$ is:

  Σ consists of:

$$\bot \leftarrow abduced(tea), not\ tea$$
$$\bot \leftarrow abduced(coffee), not\ coffee$$

Γ consists of:

$drink \leftarrow tea$
$drink \leftarrow coffee$

$expect(tea)$
$expect(coffee)$
$expect\_not(coffee) \leftarrow blood\_pressure\_high$

$0\,\{abduce(tea), abduce(coffee)\}\,1$
$coffee \prec tea \leftarrow sleepy$

$\bot \leftarrow tea, not\ expect(tea)$
$\bot \leftarrow coffee, not\ expect(coffee)$
$\bot \leftarrow tea, not\ expect\_not(tea)$
$\bot \leftarrow coffee, not\ expect\_not(coffee)$

$\bot \leftarrow tea, not\ abduced(tea)$
$\bot \leftarrow coffee, not\ abduced(coffee)$

$\bot \leftarrow coffee \prec tea \leftarrow expect(coffee), not\ expect\_not(coffee), tea, not\ coffee$

$\bot \leftarrow 0\,\{abduce(tea), abduce(coffee)\}\,1, count([abduce(tea), abduce(coffee)], N), N < 0$
$\bot \leftarrow 0\,\{abduce(tea), abduce(coffee)\}\,1, count([abduce(tea), abduce(coffee)], N), N > 1$

Properties

- The correctness of the transformation γ is guaranteed by these two properties

Property 1.
Let P be a program over $\mathcal{L}$ with set of abducibles $\mathcal{A}_P$.
M is an abductive stable model with hypotheses $\Delta \subseteq \mathcal{A}_P$ of P iff
M is a valid stable model with hypotheses Δ of $\gamma(P)$

Property 2.
Let P be a program over $\mathcal{L}$ and G a goal.
$\Delta \subseteq \mathcal{A}_P$ is a strict abductive explanation for G in P iff
Δ is a strict valid explanation for G in $\gamma(P)$

A posteriori preferences

- A priori preferences are often insufficient to express final choices
- Sometimes we are able to enact certain choices only after looking at their consequences
- These consequences are only available after model generation
- Only after the relevant models are computed can we reason about which consequences, or other features of the model, are determinant for the final choice, ie the absolute quality of the model, and its quality compared to other models

Additional Consequences of Abductive Hypotheses

- Consider the simple abductive logic program:

$$c \leftarrow a$$
$$1\ \{a, b\}\ 1$$
$$expect(a)$$
$$expect(b)$$

  Two abductive stable models can be derived:
  - $M_1 = \{expect(a), expect(b), a, c\}$
  - $M_2 = \{expect(a), expect(b), b\}$

- Suppose that c is an unwanted literal
  Hence, we would like to prefer models without c
- For this particular program we could simply state $b \prec a \leftarrow c$
  However, we should add a similar rule for every possible combination of abducibles which implies c
- To express this meta-preference it is convenient to simply look at the computed models, and prefer a posteriori the ones where c is not present

Utility Theory

- Quantitative decision making mechanisms can be employed a posteriori for enacting the final choice
- Simple Methodology:
  - Associate every model with an expected utility value computed upon the hypotheses assumed in the model
  - Prefer a posteriori by choosing those models with greater expected utility

Example: Claire Goes on Holiday!

- Claire is spending a day on the beach and must decide what means of transportation to adopt:
  - Car is faster, but there may be traffic jams
  - Train takes a lot of time, but it is more environment friendly
  - Bus, but it is really a pain..
- There are several utility factors for Claire in this situation: getting to the beach fast, the level of comfort, being more environment friendly, etc.

- Claire's Holiday, never by bus

$$
\begin{array}{ll}
go\_to(beach) \leftarrow car & expect(car) \\
go\_to(beach) \leftarrow train & expect(train) \\
go\_to(beach) \leftarrow bus & expect(bus) \\
1\ \{car, train, bus\}\ 1 & \\
 & car \prec bus \\
probability(traffic\_jam, 0.7) \leftarrow hot & train \prec bus \\
probability(not\ traffic\_jam, 0.3) \leftarrow hot & \\
 & hot \\
utility(comfort, 10) & \\
utility(stuck\_in\_traffic, -8) & \\
utility(wasting\_time, -4) & \\
utility(environment\_friendly, 3) &
\end{array}
$$

- For each scenario compute the expected utility:

  Car scenario
  Exp.Utility = Utility(Comfort) * Prob(¬ Traffic Jam) + Utility(Stuck in Traffic) * Prob(Traffic Jam) = 10 * 0.3 + 0.7 * -8 = -2.6

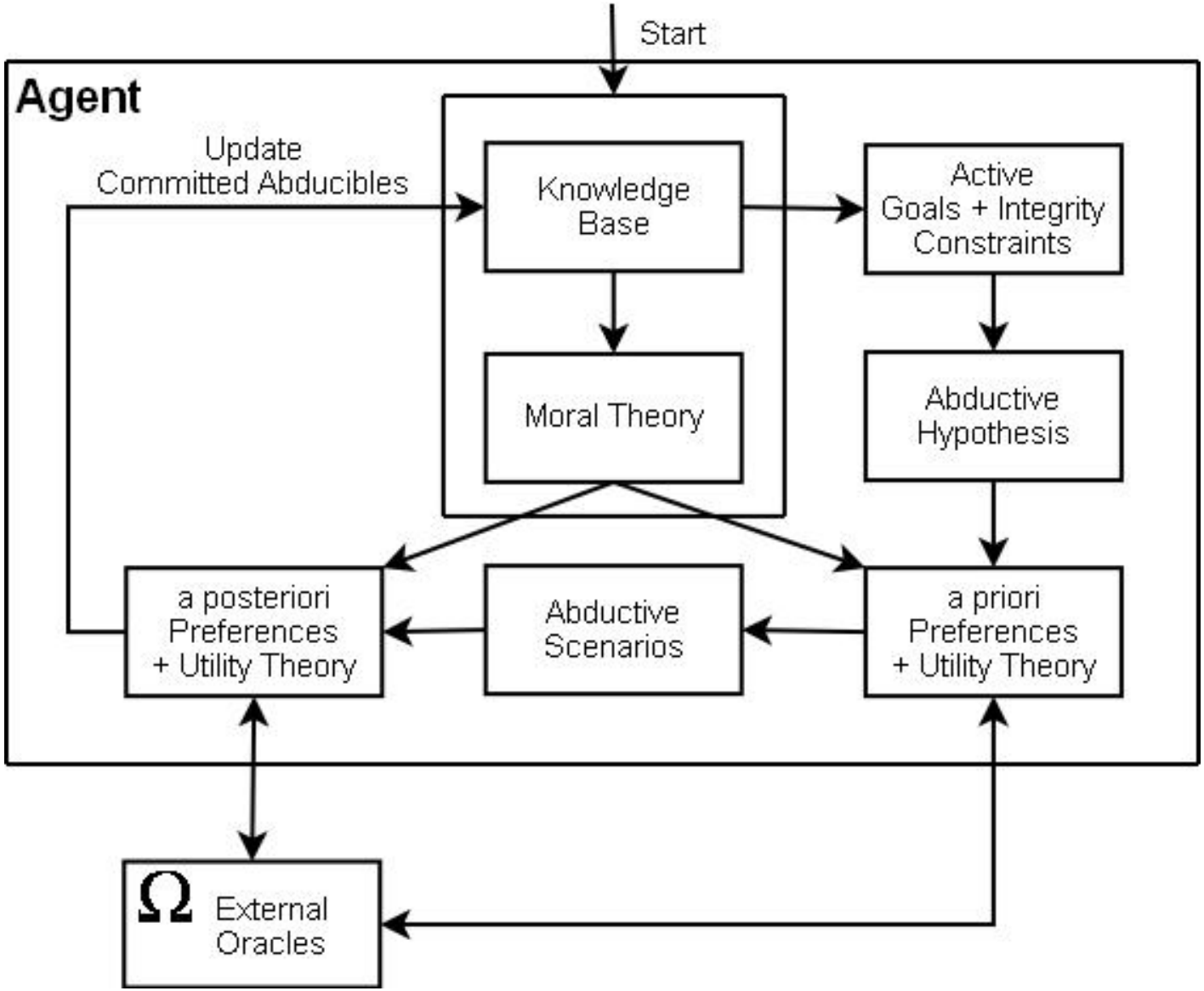
  Train scenario
  Exp. Utility = Utility(Wasting Time) + Utility(Environment Friendly) = -4 + 3 = -1

* Enacting preferential reasoning over the utilities computed for each model has to be performed after the scenarios are available, with an a posteriori meta-reasoning over the models and their respective utilities

Oracles

- When faced with several hypotheses for which one cannot prefer (neither a priori nor a posteriori), then additional information can be acquired
- This can be done by:
  - performing an experiment or
  - asking an external system, an oracle
- Information provided by the oracle has to be incorporated into the abductive logic program and a new reasoning cycle ensues
- This process can be iterated as many times as necessary to yield a final choice
- A possible termination condition is that of reaching a fixed point:
  - there are no further experiments to perform and
  - the additional information doesn't change the scenario obtained in the previous step
- Otherwise, preferences may be updated

Abductive Agent Cycle



Medical Diagnosis

- A patient shows up at the dentist with signs of pain upon teeth percussion. The expected causes for the observed signs are:
  - Periapical lesion
  - Horizontal Fracture of the root and/or crown
  - Vertical Fracture of the root and/or crown
- Producing a diagnosis consists in deriving abductive hypotheses to explain the query *percussion_pain*
- A medical knowledge base can point to these diagnoses, and also to additional consequences expected when assuming each scenario

$percussion\_pain \leftarrow periapical\_lesion$
$percussion\_pain \leftarrow fracture$
$fracture \leftarrow horizontal\_fracture$
$fracture \leftarrow vertical\_fracture$

$radiolucency \leftarrow periapical\_lesion$
$tooth\_mobility \leftarrow horizontal\_fracture$
$elliptic\_fracture\_trace \leftarrow horizontal\_fracture$
$decompression\_pain \leftarrow vertical\_fracture$

- The literals *periapical_lesion*, *horizontal_fracture* and *vertical_fracture* are abducibles
- There are three possible explanatory scenaria for *percussion_pain*
- We can attempt to disprove/confirm some of the hypotheses by performing experiments on the expected consequences of each scenaria

- The experiments and their results are derived from a secondary reasoning process over a knowledge base of experiments (example):

$$expect(xray) \leftarrow possible(radiolucency)$$
$$expect(xray) \leftarrow possible(elliptic\_fracture\_trace)$$
$$expect\_not(xray) \leftarrow radiotherapy\_patient$$

- Expected consequences of each abduced scenario are asserted as possibilities in this knowledge base

Medical Diagnosis

- The choice of what experiment to perform can be realized by an abductive process, in which the experiments are now the abducibles
- In this way it is possible to declaratively specify preferences between possible experiments (e.g. attending to the patient's comfort, economic possibilities, etc.)
- After choosing which oracle to consult, where observation instruments can be an oracles, the experiment is performed

- Confirming or disconfirming an expected consequence can introduce new constraints in the knowledge base, for instance, the constraint:

$$\bot \leftarrow \mathit{tooth\_mobility}$$

  can be derived after confirmation that the tooth has no mobility
- Such new constraints will possibly change the scenaria which were previously generated

Moral Decision Making

- Studies on morality
- Interdisciplinary perspectives
  - Philosophy:
    virtue ethics, utilitarianism/consequentialism,
    deontological principles/nonconsequentialism, etc.
  - Science:
    primatology, cognitive sciences, neuroscience, artificial intelligence, etc.
- Moral decision making involves conjuring possible courses of action, evaluating their consequences according to moral principles, and opting for a preferred outcome

Computational Study on Morality

- Several names: machine ethics, machine morality, artificial morality, computational morality
- Two purposes:
  - To understand morality better, from the computational point of view
  - To equip artificial agents with the capability of moral decision making

Sophie's Choice

One of the most discussed cases where the same moral precept gives rise to conflicting obligations is taken from William Styron's "Sophie's Choice"

Sophie and her two children are at a Nazi concentration camp. A guard confronts Sophie and tells her that one of her children will be allowed to live and one will be killed.

But it is Sophie who must decide which child will be killed. Sophie can prevent the death of either of her children, but only by condemning the other to be killed.

The guard makes the situation even more excruciating by informing Sophie that if she chooses neither both will be killed.

With this added factor, Sophie has a morally compelling reason to choose one of her children. But for each child, Sophie has an apparently equally strong reason to save him or her.

Thus the same moral precept gives rise to conflicting obligations.

- This can be formalized as:

$$\mathcal{A}_P = \{letting\_both\_die, kill(child\_1), kill(child\_2), flip\_a\_coin\}$$

$$expect(kill(child\_1))$$
$$expect(kill(child\_2))$$
$$expect(flip\_a\_coin)$$
$$expect(letting\_both\_die)$$

$$on\_observe(decide) \leftarrow sophie\_choice$$

$$decide \leftarrow letting\_both\_die, not\ kill, not\ flip$$
$$decide \leftarrow choose, not\ flip$$
$$decide \leftarrow flip$$

$$choose \leftarrow kill(child\_1), not\ kill(child\_2)$$
$$choose \leftarrow kill(child\_2), not\ kill(child\_1)$$

$$kill \leftarrow kill(child\_1)$$
$$kill \leftarrow kill(child\_2)$$

$$flip \leftarrow flip\_a\_coin$$

$$expect\_not(kill(C)) \leftarrow special\_reason(C)$$

$$expect\_not(flip\_a\_coin) \leftarrow special\_reason(child\_1), not\ special\_reason(child\_2)$$
$$expect\_not(flip\_a\_coin) \leftarrow special\_reason(child\_2), not\ special\_reason(child\_1)$$

$$die(2) \leftarrow letting\_both\_die$$
$$die(1) \leftarrow choose$$
$$die(1) \leftarrow flip$$

$$pr(feel\_guity, 1) \leftarrow kill(child\_1)$$
$$pr(feel\_guity, 1) \leftarrow kill(child\_2)$$
$$pr(feel\_guity, 0.5) \leftarrow flip\_a\_coin$$

$$Ai \ll Aj \leftarrow provable(die(N), Ai), provable(die(K), Aj), N < K$$
$$Ai \ll Aj \leftarrow provable(pr(feel\_guity, Pi), Ai), provable(pr(feel\_guity, Pj), Aj), Pi < Pj$$

$Ai \ll Aj$ stands for Ai abductive solution set being preferred to set Aj

- The strict abductive explanations for the goal $?-decide$ are:

  $\Delta_1 = \{letting\_both\_die, not\ kill(child\_1), not\ kill(child\_2), not\ flip\_a\_coin\}$

  $\Delta_2 = \{kill(child\_1), not\ kill(child\_2), not\ flip\_a\_coin\}$

  $\Delta_3 = \{kill(child\_2), not\ kill(child\_1), not\ flip\_a\_coin\}$

  $\Delta_4 = \{flip\_a\_coin\}$

- It is possible for Sophie to decide:
  - to let both of her children die
  - to choose one of her own volition
  - to flip a coin to decide
- In the next stage, a posteriori preferences are taken into account to filter out the less preferred abductive solutions

- *A posteriori preference*
  - Sophie's final decision is to flip a coin since only one child will die and she will feel less guilty
  - In case some special reason for a single child (eg. *child_1*) is entered, then the expectation to the contrary of killing *child_1* and of flipping a coin are held

    Only two strict abductive explanations are available for Sophie: one including *letting_both_die* and one including $kill(child\_2)$

    Then by applying a posteriori preference, the first one is ruled out: Sophie's final decision is to kill the *child_2*

Implementation

- XSB-XASP Interface
- Top-Down Proof Procedure
- Computation of Abductive Stable Models
- Inspection Points
- A Posteriori Choice Mechanisms

XSB-XASP Interface

- Several stable implementations have been developed and refined over the years for logic programs
- The XSB Prolog system is one of the most sophisticated, efficient and versatile among these
  - It implements program evaluation following the Well-Founded Semantics (WFS) for normal logic programs
- The semantics of Stable Models has become a cornerstone for defining some of the most important results in logic programming
  - However, its lack of some important properties of the WFS semantics, like relevancy and cumulativity, somewhat reduces its applicability in practice, namely regarding abduction

- The XASP interface has been included in XSB Prolog as a practical programming interface to Smodels
- The XASP system allows to effectively combine 3-valued with 2-valued reasoning, to get the best of both worlds
  - This is achieved by using Smodels to compute the stable models of the so-called residual program
  - This integration allows one to maintain the relevance property for queries over programs
- To avoid computation of models of the complete program and its abducibles; and obtain just the relevant abductive solution subset; on which to enact a posteriori preferences,
  we perform a preliminary computation of the relevant residual program for a query; that also takes into account the consequences of abduction we wish to inspect

Top-Down Proof Procedure

- Our implementation aims for query-driven evaluation of abductive stable models
- Computation of such models is performed in two stages
- First, XSB computes the Well-Founded Model (WFM) of the program wrt. a given query:
  - it collects all expected abducibles to prove the query
  - it enforces integrity constraints by querying *not* $\bot$ in conjunction with the query

- At this stage, the dynamically collected relevant abducibles must not be assumed neither true nor false:
  - They must come up undefined in the derivation tree
  - Which is achieved by coding considered abducibles in this thus

$$consider(A) \leftarrow expect(A), not\ expect\_not(A), abduce(A)$$

$$abduce(A) \leftarrow not\ abduce\_not(A)$$
$$abduce\_not(A) \leftarrow not\ abduce(A)$$

Computation of Abductive Stable Models

- In the second stage, Smodels is used to compute abductive stable models of the residual program
- Determination of relevant abducibles is performed beforehand by examining the residual program for ground literals which are argument to *consider* clauses
- Relevant preference rules are identified beforehand as well by querying for any such rules involving pairs of considered relevant abducibles
- After both the relevant abducibles and preference rules are determined, a variation of the procedural transformation Γ is applied, every encoded clause being sent to the XASP store for Smodels evaluation

Inspection Points

- Inspection points are implemented by adapting the evaluation of derivation subtrees falling under *inspect* literals
- Considered abducibles evaluated under *inspect* subtrees are coded as:

$$consider(A) \leftarrow abduced(A)$$

$$abduced(A) \leftarrow not\ abduced\_not(A)$$
$$abduced\_not(A) \leftarrow not\ abduced(A)$$

- All the *abduced* predicates are collected during the computation of the residual program
- To be accepted, a model must satisfy the condition that abducible A belongs to it every time $abduced(A)$ does

A Posteriori Choice Mechanisms

- If just a single model emerges from the computation of the abductive stable models, goal evaluation can successfully terminate
- When multiple models still remain, then a posteriori choice mechanisms may be employed
  - The implementation supports a hook which the user can opt for running specific code for the desired a posteriori mechanism
- In addition, we have also implemented our framework in Neg-Abdual
  - An implemented XSB-Prolog system combining constructive negation into the Well-Founded Semantics with Abduction called Abdual
- All our examples have been tested with success in both these systems

Conclusions

- A priori preferences over abductive logic programs are an important tool for knowledge representation in modeling different kinds of choice situations
- The broadening of our research direction to incorporate a posteriori preferences is a major topic of interest for research into prospective agents
  - These are agents that can not only consider their immediate context, but also exert a modicum of lookahead and meta-reasoning over available scenarios, using a combination of qualitative and quantitative criteria for decision making

- We have shown the importance of using selected consequences to constrain and condition preference handling itself, and how computation of models can be made to include specific side-effects of abduction for inspection
- Though we did not consider the issue of imposing partial order for preferences, elsewhere we have shown this need not be so, namely by specifying different conditions for revising contradictory preferences
- We have shown the advantages of implementing our framework as a hybrid Prolog-Smodels system via the XASP package, bringing together the 2- and 3-valued communities
- On a more general note: **It appears that the practical use and implementation of abduction in knowledge representation and reasoning, by means of declarative languages and systems, has reached a point of maturity, and of opportunity for development, worthy the calling of attention of a wider community of potential practitioners**