HORN CLAUSE LOGIC

[Lloyd, Foundations of Logic Programming, Springer]

- a generic clause

$$A_1 \vee \dots \vee A_k \vee \neg B_1 \vee \dots \vee \neg B_n$$

can equivalently be represented in the form

$$\underbrace{A_1, \dots, A_k}_{\text{conclusions}} \leftarrow \underbrace{B_1, \dots, B_n}_{\text{hypotheses}}$$

to be "logically read" as

$$(B_1 \wedge \quad \wedge B_n) \supset (A_1 \vee \dots \vee A_k)$$

- Horn clauses are clauses with at most one conclusion $(k \leq 1)$

  - definite (program) clauses $\quad k=1$

    $$A \leftarrow B_1, \dots, B_n$$

    - unit clause (assertion) $\quad k=1, n=0$

      $$A \leftarrow$$

  - goal (negative) clauses, queries $\quad k=0$

    $$\leftarrow B_1, \dots, B_n$$

    - empty clause $\quad k=0, n=0$

      $$\square$$

PURE LOGIC PROGRAMMING

- a program is a set of definite clauses

the goal $\leftarrow B_1, \ldots B_n$

$$\forall X_1, \ldots, X_k \ (\neg B_1 \vee \ldots \vee \neg B_n)$$

is the negation of

$$\boxed{\exists X_1, \ldots, X_k \ (B_1 \wedge \ldots \wedge B_n)}$$

- formulas which can be proved are restricted to this form
  axioms are restricted to Definite Horn Clauses
  - Horn clause theories are always consistent
  - there exists a complete resolution strategy which is essentially nondeterministic search
  - there exists a ~~can~~ "canonical" Herbrand model
  - a goal refutation returns a substitution (computed answer)

HERBRAND UNIVERSE, BASE AND MODELS

- **Herbrand Universe** — the set of all the ground terms
- **Herbrand Base** — the set of all the ground atoms
  - if the language has no constant symbols, we add an arbitrary constant symbol

- **Herbrand interpretation**
  - the domain is the Herbrand Universe
  - constants, functions ~~and~~ predicates are interpreted "syntactically"
  - a Herbrand interpretation can be represented as any subset of the Herbrand Base (the set of ground atoms which are true!)

- **Herbrand model**

  a Herbrand interpretation I such that all the clauses in C are true in I

SLD-RESOLUTION

P logic program

$N = \leftarrow A_1, \dots, A_n$ goal

$C = A \leftarrow B_1, \dots, B_m$ clause in P

If for some i, A_i and A are unifiable with mgu ϑ:
the new goal is

$$N' = \leftarrow (A_1, \dots, A_{i-1}, B_1, \dots, B_m, A_{i+1}, \dots, A_n)\vartheta$$

resolvent of N and C

- selection of the atom A_i in the goal (selection rule)
- unification of A_i with the clause head $A \rightarrow \vartheta$ (if successful)
- replacement of procedure call A_i with procedure body $B_1, \dots, B_m$
- application of ϑ to the resulting clause (parameter passing in both directions)

- $\boxed{\text{SLD-derivation of } P \cup \{N\}}$

a sequence of goals $N_0 = N, N_1, \dots$

a sequence of variants of clauses in P $C_0, C_1, \dots$

a sequence of substitutions $\vartheta_0, \vartheta_1, \dots$

- N_{i+1} is a resolvent of N_i, C_i with mgu C_i
- C_i does not share variables with $N_0, C_0, \dots, C_{i-1}$
  (standardization apart)

- if $N_i = \square$, the derivation terminates and is called an SLD-refutation

SELECTION RULE

- the policy chosen to select the atom in the goal
  - may depend on the history of the derivation
  - we will almost always assume the leftmost selection rule (the rule of PROLOG)

SLD - TREE

- given a selection rule, we can still have nondeterminism in the construction of an SLD- derivation
  - which (variant of) clause is chosen whose head unifies with the selected atom
- the set of all possible SLD-derivations (for a given selection rule) are represented by an SLD-tree
  - the root is N
  - each node is a goal
  - the successors of a node are all its resolvents with variants of clauses of P, whose heads unify with the selected literal
- the search rule specifies how to visit the SLD-tree
  - whenever necessary, we assume a "fair" search rule

CORRECTNESS OF SLD-RESOLUTION

logic program P
goal $N = \leftarrow A_1, \ldots, A_k$

- let us assume there exists an SLD-refutation of $P \cup \{N\}$, whose sequence of substitutions is $\vartheta_0, \ \ , \vartheta_n$

- the general correctness result for the resolution method tells us that

$$\exists X_1, \ldots, X_m \, (A_1 \wedge \ldots \wedge A_k)$$

is logical consequence of P

- a stronger result holds

the universal closure of the formula

$$A_1 \wedge \ldots \wedge A_k) \vartheta_0 \vartheta_1 \ldots \vartheta_n$$

is logical consequence of P

COMPUTED ANSWER SUBSTITUTION

the restriction to the variables occurring in N of the composition of substitutions $\vartheta_0 \vartheta_1 \ldots \vartheta_n$

- the stronger correctness result gives a semantic (model-theoretic) meaning to computed answers

THE PROOF-THEORETIC PROGRAM DENOTATION

- sometimes called "operational semantics"
  - program P
  - Herbrand Base B_P

$$O_P = \{ A \in B_P \mid P \cup \{\leftarrow A\} \text{ has an SLD-refutation} \}$$

- success set
- set of refutable ground atoms

- it is a Herbrand interpretation
  - is it also a model?

MODEL THEORY (DECLARATIVE SEMANTICS)

- the set of Herbrand interpretations partially ordered by set inclusion is a complete lattice

- a continuous operator from Herbrand interpretations to Herbrand interpretations

$$T_P \quad 2^{B_P} \to 2^{B_P}$$

$$T_P(I) = \left\{ A \in B_P \;\middle|\; \begin{array}{l} \bullet\ A \leftarrow A_1, \dots, A_n \text{ is a ground instance of a clause in } P \\ \bullet\ \{A_1, \dots, A_n\} \subseteq I \end{array} \right\}$$

(immediate consequences operator)

- $\boxed{\text{Herbrand models of } P \text{ are pre-fixpoints of } T_P}$

- every logic program P has a Herbrand model M_P with the following properties
  - M_P is the least Herbrand model of P
  - M_P is the least fixpoint of T_P
  - $M_P = T_P \uparrow \omega$
  - M_P is the intersection of all the Herbrand models
  - M_P is the set of all the ground atoms which are logical consequences of P
  - $M_P = O_P$

THE MODEL-THEORETIC PROGRAM DENOTATION

M_P

the least Herbrand model of P

the least fixpoint of T_P (fixpoint or denotational semantics)

COMPLETENESS OF SLD-RESOLUTION

- correct answer substitution

  P program
  $N = \leftarrow A_1, \dots, A_n$ goal

  ϑ is a correct answer substitution for $P \cup \{N\}$ if
  - domain(ϑ) contains only variables occurring in N
  - $\forall (A_1 \wedge \dots \wedge A_n)\vartheta$ is a logical consequence of P

- (a corollary of the correctness theorem)

  every computed answer substitution is a correct answer substitution

- the completeness theorem (Clark, 1979)

  P program N goal

  if ϑ is a correct answer substitution for $P \cup \{N\}$, there exists a computed answer substitution ϑ' for $P \cup \{N\}$, such that $N\vartheta$ is an instance of $N\vartheta'$

  for every correct answer substitution there exists a "more general" computed answer substitution

  - a weaker consequence

    if $P \cup \{N\}$ is inconsistent, there exists an SLD-refutation

- independence from the selection rule

  computed answers do not depend upon the selection rule
  - other properties (such as finite failures) do depend

THE S-semantics APPROACH

a survey with extensive list of references

[Bossi, Gabbrielli, Levi, Martelli – The S-semantics approach: theory and applications, Journal of Logic Programming, 1994]

- inadequacy of the standard declarative semantics
  - $M_P = O_P = F_P$
    - least Herbrand model
    - success set
    - fixpoint semantics
  - one of the most advertised properties of logic programs
- the declarative semantics (logic denotation) does not capture relevant computational properties, such as computed answers
  - the correctness and completeness theorems related to correct and computed answers are stronger than the equivalence theorem $O_P = M_P$
  - computed answers are just what makes Horn clause logic a programming language
- we need to look at logic languages as "standard" programming languages

WHICH SEMANTICS

the answer depends on

- what do we need the semantics for?
  - specification for the language implementation
  - to allow the user to understand the meaning of his/her programs
  - as basic semantics for semantics-based tools (analysis, verification, transformation, interpreter and compiler generation, ...)
- which computational properties we want to model (observables)
  - success termination
  - computed answers
  - intermediate (partial) computed answers
  - procedure calls (call patterns)
  - SLD-trees
- some observables are clearly more abstract than other
- some observables are more adequate to a specific use of the semantics

conclusion

there exists no such a thing as <u>the</u> semantics

OBSERVABLES AND EQUIVALENCES

- the starting point (the most concrete semantics) is the proof-theoretic one, i.e. SLD-trees
- an <u>observable</u> α is any property which can be observed in an SLD tree (a formal definition later)
  - SLD-trees, SLD-derivations, resultants, call patterns, partial answers, computed answers, finite failures, ...
- the choice of the observable α induces an <u>equivalence relation on programs</u>

$P_1 \cong_\alpha P_2$ iff P_1 and P_2 ~~cannot~~ are not ~~be~~ distinguishable by any observation

$W_P^\alpha = \{ \langle G, \sigma \rangle \mid \sigma$ is the value of the observable α in the SLD-tree of goal $G \}$

$P_1 \cong_\alpha P_2$ iff $W_{P_1}^\alpha = W_{P_2}^\alpha$

- for every goal G the observations in P_1 and P_2 are the same

- <u>example</u>

  the observable "success" α_1

$P_1 \cong_{\alpha_1} P_2$ iff

$\forall$ goal G. $\sigma_{\alpha_1}^{P_1}(G) = \sigma_{\alpha_1}^{P_2}(G)$

- the value of the observable α_1

$$\sigma_{\alpha_1}^{P}(G) = \begin{cases} \text{yes, if } G \longrightarrow_P \blacksquare \\ \text{no, otherwise} \end{cases}$$

OBSERVABLES AND DENOTATIONS

P program

$[\![P]\!]$ a denotation of P ("semantics")

- the denotation is <u>correct</u> w.r.t. the observable α, if

$$[\![P_1]\!] = [\![P_2]\!] \rightarrow P_1 \cong_\alpha P_2 \qquad \forall P_1, P_2$$

  - an essential property:
    a semantics which identifies programs which have a different behaviour w.r.t. α is useless if one wants to reason about the observable α

- the denotation is <u>minimal</u> w.r.t. the observable α, if

$$P_1 \cong_\alpha P_2 \rightarrow [\![P_1]\!] = [\![P_2]\!] \qquad \forall P_1, P_2$$

- if a denotation is correct and minimal w.r.t. α, then the equivalence relation induced by the denotation is the same as the equivalence relation induced by the observable. the denotation is the best (most abstract) ~~reason~~ in
- the set of correct denotations
  - a correct non minimal denotation might contain too many irrelevant details, which distinguish equivalent programs and might result in a useless more complex reasoning about the observable.

- if a denotation is correct w.r.t. an observable α_1 it is also correct for any "more abstract" observable α_2

COMPOSITIONALITY

- an important property of denotations, which allows one to reason on the properties of a program by reasoning on the properties of the (syntactic) program components

- there exists an omomorphism between syntax and semantics

  $\forall$ ~~any~~ f syntactic operator

  $\exists$ F . semantic operator

  $$[\![f(A_1, \dots, A_n)]\!] = F([\![A_1]\!], \dots, [\![A_n]\!])$$

  - the typical definition style of denotational semantics

  - compositionality can typically get lost when taking the least fixpoint

A TRADITIONAL VIEW OF THE SYNTACTIC OPERATORS OF LOGIC LANGUAGES

goal $?- A_1, \dots, A_n$

definite clauses
$$\begin{cases} H_1 :- B_1^1, \dots, B_1^{n_1}. \\ \vdots \\ H_m :- B_m^1, \dots, B_m^{n_m}. \end{cases}$$

- program = set of definite clauses = set of procedure declarations
- $,$ (AND) the mechanism to syntactically compose procedure calls
- goal = a composition of procedure calls
- $.$ (OR) the mechanism to syntactically compose procedure declarations

  a program is a conjunction of clauses
  - the operator . is called OR, because it is used to represent a disjunction in the body of a single clause defining a procedure

  $A :- B, C.$ $\quad$ $A :- (B \wedge C) \vee (D \wedge E)$
  $A :- D, E.$

- which compositional properties

  procedural compositionality : from the denotation of a procedure to the denotation of a procedure call

  - AND-compositionality : from the denotation of a set of procedure calls to the denotation of their AND-composition (goal or clause body)

  OR-compositionality : from the denotations of two sets of clauses to the denotation of the OR-composition of the two sets.

PROPERTIES OF THE LEAST HERBRAND MODEL

- M_P is both correct and minimal w.r.t. the observable "success"

$$\forall P_1, P_2 \qquad M_{P_1} = M_{P_2} \iff \mathcal{W}_{P_1}^{\alpha_s} = \mathcal{W}_{P_2}^{\alpha_s}$$

  - two programs have the same least Herbrand model if and only if they have the same set of refutable goals

- procedural compositionality holds
  - a procedure call succeeds, if it has an instance in the denotation of the procedure (i.e. in M_P)
- AND-compositionality does not hold
  - the success behaviour of the goal ?- A,B cannot be predicted from the success behaviours of A and B
- let us try with another observable

  "ground instances" of computed answer substitutions

$$\alpha_2 = \left\{ \langle G, \vartheta' \rangle \;\middle|\; \begin{array}{l} G\vartheta' \text{ is ground}, \\ G \xrightarrow{\vartheta}_P \Box, \\ G\vartheta' \text{ is an instance of } G\vartheta \end{array} \right\}$$

M_P AND GROUND COMPUTED ANSWERS

- M_P is both correct and minimal w.r.t. ground instances of computed answers ($\approx_2$)

- ~~mo~~ procedural compositionality holds

$$[\![A]\!]_{M_P} = \{ \langle A, \vartheta \rangle \mid \exists B \in M_P, \vartheta' = mgu(A,B) \text{ restricted to the variables in } A, A\vartheta \text{ is a ground instance of } A\vartheta' \}$$

AND-compositionality holds

$[\![A, G]\!]_{M_P}$ can be derived from $[\![A]\!]_{M_P}$ and $[\![G]\!]_{M_P}$

- procedural and AND-compositionalities can be combined in a single theorem, which tells us that the observation for any goal G can be predicted by "executing" the goal in the denotation M_P

- goal-compositionality (condensing) theorem

$\forall G = ?- A_1, \ldots, A_n$ goal

$\langle G, \vartheta \rangle \in \mathcal{W}_P^{\approx_2}$ observation in the SLD-tree

$\Updownarrow$

$\exists \{B_1, \ldots, B_n\} \subseteq M_P$ prediction from M_P

$\vartheta' = mgu((A_1, \ldots, A_n), (B_1, \ldots, B_n))$

$G\vartheta$ is a ground instance of $G\vartheta'$

- $\approx_1$ (mucn) and $\approx_2$ (ground computed answers) define exactly the same equivalence relation
  - M_P can better be considered a denotation for $\approx_2$, because of the compositionality properties

- OR-compositionality does not hold neither for $\approx_1$ nor for $\approx_2$
  - it is necessary only when modular reasoning is required

ANOTHER PROOF-THEORETIC CONSTRUCTION OF M_P

- the proof-theoretic (operational) characterization of M_P was the success set

$$\mathcal{O}_P = \{ A \in B_P \mid \ ?\text{- } A \text{ has an SLD refutation in } P \}$$

- there exists another (equivalent) construction given in terms of the observable α_2
  - collecting all the observations for most general atomic goals
    - procedure call, with no constraints on the inputs
  - and applying the computed observations to the initial goal)

$$\mathcal{O}_P = \{ A \in B_P \mid A \text{ is a ground instance of } p(X_1,\dots,X_n)\vartheta,\ ?\text{- } p(X_1,\dots,X_n) \xrightarrow[P]{\vartheta} \Box \}$$

- this property holds for a large class of observables and will allow us to derive the proof-theoretic characterization of the denotation from the observable.

LEAST HERBRAND MODEL AND COMPUTED ANSWERS

- the observable computed answers

$$\alpha_3 = \{ \langle G, \vartheta \rangle \mid G \xrightarrow[\;3\;]{\vartheta} \square \}$$

- M_P is not correct w.r.t. α_3

$$M_{P_1} = M_{P_2} \not\Rightarrow \mathcal{W}_{P_1}^{\alpha_3} = \mathcal{W}_{P_2}^{\alpha_3}$$

Counterexample

P_1

p(a).
q(x).

P_2

p(a).
q(x).
q(a).

$$M_{P_1} = M_{P_2} = \{ p(a), q(a) \}$$

$\mathcal{W}_{P_1}^{\alpha_3} \neq \mathcal{W}_{P_2}^{\alpha_3}$, because the goal ?- q(Y) has different answers in P_1 and P_2

- ε in P_1
- ε and $\{Y \leftarrow a\}$ in P_2

- the problem might be related to the fact that M_P, being a subset of the Herbrand Base, does not properly describe the behaviour of non-ground goals

- a different denotation, defined on non-ground interpretations

FROM M_P TO THE C-SEMANTICS

$$M_P = O_P = \{ A \mid A \text{ is a ground instance of } p(X_1, \dots, X_n)\vartheta,\ ?\text{-}\ p(X_1, \dots, X_n) \xrightarrow{\vartheta}_P \blacksquare \}$$

observable "ground instances of computed answers"

$$O_P^c = \{ A \mid A \text{ is an instance of } p(X_1, \dots, X_n)\vartheta,\ ?\text{-}\ p(X_1, \dots, X_n) \xrightarrow{\vartheta}_P \blacksquare \}$$

observable "instances of computed answers" = "correct answers"

- O_P^c is not a Herbrand interpretation
  - an non ground atom in O_P^c stands for its equivalence class w.r.t. variance
- the declarative view of O_P^c
  - the set of atomic logical consequences
- O_P^c is correct w.r.t. "correct answers" but is not correct w.r.t. computed answers

P_1

```
p(a).
q(X).
```

P_2

```
p(a).
q(X).
q(a).
```

$$O_{P_1}^c = O_{P_2}^c = \{ p(a), q(X), q(a) \}$$

- in order to get a correct denotation w.r.t. computed answers, we have to get rid of instances in the definition of the proof-theoretic denotation

BOTTOM-UP FIXPOINT CONSTRUCTION OF THE DENOTATION

- as in the case of M_P, we can give an equivalent definition of O_P^S as least fixpoint of an immediate consequences operator

$$O_P = \{ A \mid A \text{ is a ground instance of } p(X_1,\dots,X_n)\vartheta,\ ?\!-\,p(X_1,\dots,X_n) \xrightarrow{\vartheta}_P \Box \}$$

$$T_P(I) = \{ A \mid A :\!-\, B_1,\dots,B_n \text{ is a ground instance of a clause in } P,\ \{B_1,\dots,B_n\} \subseteq I \}$$

$$O_P^S = \{ A \mid A = p(X_1,\dots,X_n)\vartheta',\ ?\!-\,p(X_1,\dots,X_n) \xrightarrow{\vartheta'}_P \Box \}$$

$$T_P^S(I) = \{ A \mid A' :\!-\, B_1,\dots,B_n \text{ is a clause in } P,\ \{B_1',\dots,B_n'\} \subseteq I,\ \vartheta = mgu((B_1,\dots,B_n),(B_1',\dots,B_n')),\ A = A'\vartheta \}$$

- I is a set of non-ground atoms (modulo variance)
- we use the original clauses and we compute with most general unifiers, rather than taking instances
  - as we do in SLD-resolution

- T_P^S is continuous

$$O_P^S = T_P^S \uparrow \omega$$

THE S-SEMANTICS

$\mathcal{O}_P = \{ A \mid A$ is a ground instance of $p(X_1, \dots, X_n)\vartheta$, $?- p(X_1, \dots, X_n) \xrightarrow{\vartheta}_P \square \}$

$\mathcal{O}_P^c = \{ A \mid A$ is an instance of $p(X_1, \dots, X_n)\vartheta$, $?- p(X_1, \dots, X_n) \xrightarrow{\vartheta}_P \square \}$

$\mathcal{O}_P^s = \{ A \mid A = p(X_1, \dots, X_n)\vartheta, \; ?- p(X_1, \dots, X_n) \xrightarrow{\vartheta}_P \square \}$

- same semantic domain of the c-semantics
- $\mathcal{O}_P^s$ is correct (and minimal) w.r.t. computed answers
- procedural compositionality holds
- AND-compositionality holds
- goal-compositionality (unfolding) theorem

$\forall \; G = ?- A_1, \dots, A_n$

$\langle G, \vartheta \rangle \in \mathcal{W}_P^{d_3}$ $\quad (G \xrightarrow{\vartheta}_P \square)$

if and only if

$\exists \; B_1, \dots, B_n \in \mathcal{O}_P^s$, $\vartheta' = mgu((A_1, \dots, A_n), (B_1, \dots, B_n))$, $G\vartheta = G\vartheta'$

prediction of answers from $\mathcal{O}_P^s$

- once we have computed the answers for most general atomic goals ($\mathcal{O}_P^s$), the answers for any goal G can be obtained, by "executing" the goal in the s-semantics

PROPERTIES OF THE S-SEMANTICS

- the declarative concept of correct answers (given in terms of all the models) has a characterization in terms of one model only (c-semantics or s-semantics)
  - this is not true for M_P

- the s-semantics is language independent
  - if we add new constant and function symbols, the s-semantics is not affected, while the least Herbrand model and the c-semantics are

- it is not true that M_P does not correctly models computed answers only for artificial uninteresting programs
  - $M_P = O_P^1$ if and only if P is language independent
    - a decidable approximation is

      allowed programs

      ground unit clauses

      - no partially evaluated data structures

      ≈ deductive data bases

- anyway, even the s-semantics is not always the best choice
  - the s-semantics is not OR-compositional

TOWARDS AN OR-COMPOSITIONAL SEMANTICS FOR COMPUTED ANSWERS

- $\mathcal{O}_P^s$ is not OR-compositional
  - the s-semantics of $P_1 \cup P_2$ cannot be derived from the s-semantics of P_1 and P_2

P_1 $\boxed{\begin{array}{l} p(x) :- r(x) \\ p(a) \end{array}}$ $\qquad P_2$ $\boxed{r(b).}$

$\mathcal{O}_{P_1}^s = \{p(a)\}$ $\qquad \mathcal{O}_{P_2}^s = \{r(b)\}$

$\mathcal{O}_{P_1 \cup P_2}^s = \{p(a), p(b), r(b)\}$

- T_P^s is OR-compositional (by construction), but $T_P^s \uparrow \omega$ is not

- OR-compositionality requires to maintain the relations among predicates
  - T_P is a function from interpretations to interpretations (T_P^s !)
  - by means of clauses

- OR-compositionality can be embedded into the definition of observational equivalence (cd = compositional computed answers)

$P_1 \cong_{cd} P_2$ if

$\forall G$ goal $\forall P$ program

$G \xrightarrow[P_1 \cup P]{\vartheta} \Box \qquad G \xrightarrow[P_2 \cup P]{\vartheta'} \Box$

$G\vartheta$ is a variant of $G\vartheta'$

TOP-DOWN CHARACTERIZATION OF OR-COMPOSITIONAL COMPUTED ANSWERS

$$\mathcal{O}_P^{\alpha_4} = \left\{ p(X_1,\dots,X_n)\vartheta :- B_1,\dots,B_n \;\middle|\; ?\,p(X_1,\dots,X_n) \overset{\vartheta}{\underset{P}{\Longrightarrow}} ?\,B_1,\dots,B_n \right\}$$

- correctness

$$\text{if } \mathcal{O}_{P_1}^{\alpha_4} = \mathcal{O}_{P_2}^{\alpha_4} \Longrightarrow P_1 \cong_{\alpha_4} P_2$$

- OR-compositionality

$$\mathcal{O}_{P_1 \cup P_2} = \mathcal{O}_{\mathcal{O}_{P_1} \cup \mathcal{O}_{P_2}}$$

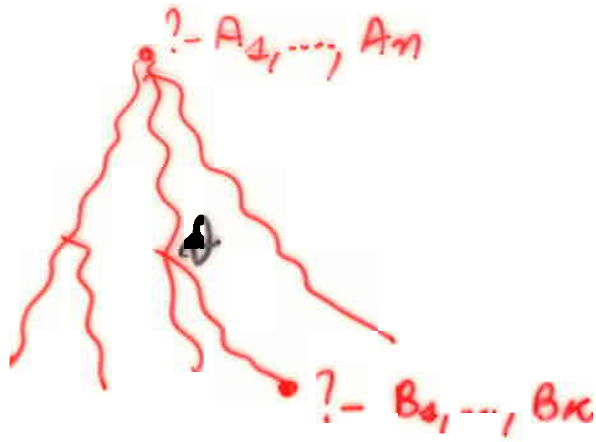
- is not minimal (and therefore not fully abstract)

$$T_P^{\alpha_4}(I) = \left\{ c \;\middle|\; \begin{array}{l} A :- B_1,\dots,B_n \in P \\ H_1,\dots,H_{s-1},\ H_s :- G_1,\dots,G_k \in I \\ \vartheta = mgu((B_1,\dots,B_s),(H_1,\dots,H_s)) \\ c = (A :- G_1,\dots,G_k,B_{s+1},\dots,B_n)\vartheta \end{array} \right\}$$

$$T_P^{\alpha_4} \uparrow \omega = \mathcal{O}_P^{\alpha_4}$$

RESULTANTS

- the same semantics which characterize computed answers in an OR-compositional way to model a different observable, i.e. resultants



- any intermediate state of an SLD-derivation can be represented by a formula

$$B_1 \wedge \ldots \wedge B_k \supset (A_1, \ldots, A_n)\vartheta$$

ϑ restriction to the initial goal of the composition of the mgu's

if the initial goal is atomic

a resultant is a Horn clause

$$A_1\vartheta :- B_1, \ldots, B_k$$

the observable α_5

THE RESULTANTS SEMANTICS

$O_P^{\alpha_4} = T_P^{\alpha_4} \uparrow \omega$ is

- correct
- minimal
- OR-compositional

wrt. α_5 (resultants)

- the goal-compositionality (condensing) theorem

  c is a resultant of $?- A_1, \dots, A_n$ in P

  iff

  $\exists \{H_1, \dots ; H_{s-1}, H_s :- B_1, \dots, B_k\} \in O_P^{\alpha_4}$

  $\vartheta = mgu((A_1, \dots, A_s), (H_1, \dots, H_s))$

  $c' = ((A_1 \wedge \dots \wedge A_m) \leftarrow B_1, \dots, B_k, A_{s+1}, \dots, A_m)\vartheta$

  c' is a variant of c

OTHER (MORE ABSTRACT) OBSERVABLES

- call patterns
  - procedure calls

  binary clauses

- partial answers
  - answers computed at intermediate steps
  - same domain of the s-semantics

THE COMMON FEATURES

- top-down definition

  collect all the observables for the goals of the form $?- p(X_1, \ldots, X_n)$

- (equivalent) bottom-up definition

- the denotations are goal independent

- the observable for a specific goal G can be determined by "executing" G in the denotation

- ~~the construction can be useful for abstract interpretation (goal independent)~~

LOGIC DENOTATION VS PROGRAM DENOTATION

- from the purely logical viewpoint we don't care about issues like observational equivalences and compositionality
  - the standard denotation is ok
- if the denotation has to be used in semantics-based program manipulation we are forced to be concerned with the more computational issues
  - program transformations should preserve at least computed answers, and therefore a denotation correct wrt computed answers
  - a program analysis aiming at establishing properties of the computed answers (e.g. aliasing or groundness analysis) should abstract a denotation correct wrt computed answers
  - a modular program analysis technique requires an OR-compositional denotation
  - a program analysis aiming at establishing properties of the procedure calls (for optimization purposes) needs a denotation correct wrt an observable which shows more internal computation details
- many different semantics containing more information than the purely logical information of the standard denotation