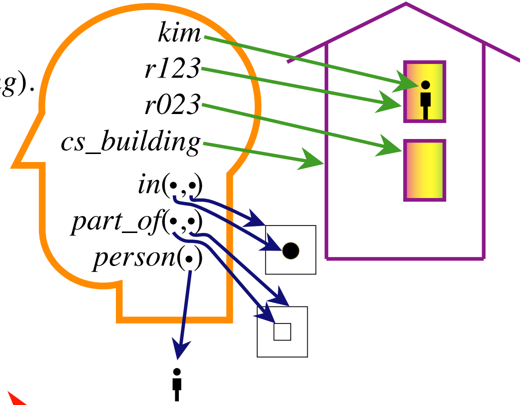
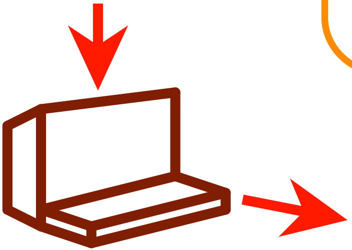
Individuals and Relations

- It is useful to view the world as consisting of individuals (objects, things) and relations among individuals.
- Often features are made from relations among individuals and functions of individuals.
- Reasoning in terms of individuals and relationships can be simpler than reasoning in terms of features, if we can express general knowledge that covers all individuals.
- Sometimes we may know some individual exists, but not which one.
- Sometimes there are infinitely many individuals we want to refer to (e.g., set of all integers, or the set of all stacks of blocks).

Role of Semantics in Automated Reasoning

in(*kim*,*r123*).
part_of(*r123*,*cs_building*).
in(*X*,*Y*) ←
    part_of(*Z*,*Y*) ∧
    in(*X*,*Z*).





in(*kim*,*cs_building*)

Features of Automated Reasoning

- Users can have meanings for symbols in their head.
- The computer doesn't need to know these meanings to derive logical consequence.
- Users can interpret any answers according to their meaning.

First Order Logic

Predicate Logic, or, First Order Logic (FOL)

- Terms as named for individuals
  - b_obama, *x*, *y*, *z, birthdate(b_obama), birthdate(x)*
- Predicates to refer to relations among different individuals
  - *PresidentOf*(*x*,y,*s*,*t*)
- Atomic formulas to state that relations between individuals exist
  - *PresidentOf*(b_obama,usa,2009,2017)
- Complex formulas via connettives and quantifier
  - $\forall x(\exists y \exists z, PresidentOf(x, usa, y, z) \rightarrow USPresident(x))$

First Order Logic

Inferences in FOL

$$\frac{\begin{array}{c}\forall x(\exists y \exists z, PresidentOf(x,\mathrm{usa},y,z) \rightarrow USPresident(x)) \\ PresidentOf(\mathrm{b\_obama},\mathrm{usa},2009,2017)\end{array}}{USPresident(\mathrm{b\_obama})}$$

$$\frac{\begin{array}{c}\textit{Tutti i siciliani sono giardinieri} \\ \textit{Barack Obama è siciliano}\end{array}}{\textit{Barack Obama è giardiniere}}$$

$$\frac{\begin{array}{c}\forall x Siciliano(x) \rightarrow Giardiniere(x) \\ Siciliano(\mathrm{b\_obama})\end{array}}{Giardiniere(\mathrm{b\_obama})}$$

KR & Logic: Reasoning in Theory and in Practice

- Decidability
  - Given a formula of the logic (and a set of premixes) are we sure that there is a finite proof to determine if the formula is true or false (or, follows from the premises
    - Decidable (e.g., PROP) vs Indecidable (e.g., FOL)
- Complexity
  - If decidable, how fast is reasoning? How many formulas can we reason about?
    - Depends on the expressiveness of the language: subsets of FOL are decidable, and faster. For example, Description Logics, used in OWL
- Approximation
  - Can we devise inference engines that use rules, i.e., quasi-logical expressions that are more efficient to deal with (possibly relaxing completeness of inferences)?
    - Logic programs (DLV, PROLOG) vs rule-based systems (iLOG, JESS, Jena rules)

KR & Logic: Expressivity // Uncertainty

- Expressivity
  - Which kind of statements can a logic express, with which kind of primitives?
    - Deontic Logics (Obligations/Permissions)
    - Epistemic Logics (Beliefs)
    - Spatial, Temporal, Spatio-temporal Logics (Topology, Directions, LTL,…)
    - Causality & Events
- Beyond 1/0: logics for reasoning about information incompleteness, uncertainty, vagueness
  - Information incompleteness: premixes are incomplete
    - Non Monotonic Logics (Negation as failure, Default, Autoepistemic)
  - Uncertainty as a first-class citizen: premixes & conclusions are uncertain
    - Multi-valued Logics, PSL (Probabilistic Soft Logic)
  - Vagueness: some concepts, e.g., "*tall*", are intrinsically vague
    - Fuzzy logics & fuzzy sets
    - Analogical reasoning & similarity

Representational Assumptions of Datalog

- An agent's knowledge can be usefully described in terms of *individuals* and *relations* among individuals.
- An agent's knowledge base consists of *definite* and *positive* statements.
- The environment is *static*.
- There are only a finite number of individuals of interest in the domain. Each individual can be given a unique name.

$\Longrightarrow$ Datalog

Syntax of Datalog

- A variable starts with upper-case letter.
- A constant starts with lower-case letter or is a sequence of digits (numeral).
- A predicate symbol starts with lower-case letter.
- A term is either a variable or a constant.
- An atomic symbol (atom) is of the form p or $p(t_1, \ldots, t_n)$ where p is a predicate symbol and t_i are terms.

Syntax of Datalog (cont)

- A **definite clause** is either an atomic symbol (a fact) or of the form:

$$\underbrace{a}_{\text{head}} \leftarrow \underbrace{b_1 \wedge \cdots \wedge b_m}_{\text{body}}$$

  where a and b_i are atomic symbols.
- **query** is of the form $?b_1 \wedge \cdots \wedge b_m$.
- **knowledge base** is a set of definite clauses.

Example Knowledge Base

$$
\begin{aligned}
& in(kim, R) \leftarrow \\
& \qquad teaches(kim, cs322) \wedge \\
& \qquad in(cs322, R). \\
& grandfather(william, X) \leftarrow \\
& \qquad father(william, Y) \wedge \\
& \qquad parent(Y, X). \\
& slithy(toves) \leftarrow \\
& \qquad mimsy \wedge borogroves \wedge \\
& \qquad outgrabe(mome, Raths).
\end{aligned}
$$

Semantics: General Idea

A semantics specifies the meaning of sentences in the language.
An interpretation specifies:

- what objects (individuals) are in the world
- the correspondence between symbols in the computer and objects & relations in world
  - constants denote individuals
  - predicate symbols denote relations

Formal Semantics

An interpretation is a triple $I = \langle D, \phi, \pi \rangle$, where

- D, the domain, is a nonempty set. Elements of D are individuals.
- ϕ is a mapping that assigns to each constant an element of D. Constant c denotes individual $\phi(c)$.
- π is a mapping that assigns to each n-ary predicate symbol a relation: a function from D^n into $\{\mathit{TRUE}, \mathit{FALSE}\}$.

Example Interpretation

Constants: *phone*, *pencil*, *telephone*.

Predicate Symbol: *noisy* (unary), *left_of* (binary).

- $D = \{$✂, ☎, ✎$\}$.
- $\phi(phone) =$ ☎, $\phi(pencil) =$ ✎, $\phi(telephone) =$ ☎.
- $\pi(noisy)$:

⟨✂⟩	*FALSE*	⟨☎⟩	*TRUE*	⟨✎⟩	*FALSE*

  $\pi(left\_of)$:

⟨✂, ✂⟩	*FALSE*	⟨✂, ☎⟩	*TRUE*	⟨✂, ✎⟩	*TRUE*
⟨☎, ✂⟩	*FALSE*	⟨☎, ☎⟩	*FALSE*	⟨☎, ✎⟩	*TRUE*
⟨✎, ✂⟩	*FALSE*	⟨✎, ☎⟩	*FALSE*	⟨✎, ✎⟩	*FALSE*

Important points to note

- The domain D can contain real objects. (e.g., a person, a room, a course). D can't necessarily be stored in a computer.
- $\pi(p)$ specifies whether the relation denoted by the n-ary predicate symbol p is true or false for each n-tuple of individuals.
- If predicate symbol p has no arguments, then $\pi(p)$ is either *TRUE* or *FALSE*.

Truth in an interpretation

A constant c denotes in I the individual $\phi(c)$.
Ground (variable-free) atom $p(t_1, \ldots, t_n)$ is

- true in interpretation I if $\pi(p)(\langle\phi(t_1), \ldots, \phi(t_n)\rangle) = \textit{TRUE}$ in interpretation I and
- false otherwise.

Ground clause $h \leftarrow b_1 \wedge \ldots \wedge b_m$ is false in interpretation I if h is false in I and each b_i is true in I, and is true in interpretation I otherwise.

Example Truths

In the interpretation given before, which of following are true?

$$
\begin{array}{l}
noisy(phone) \\
noisy(telephone) \\
noisy(pencil) \\
left\_of(phone, pencil) \\
left\_of(phone, telephone) \\
noisy(phone) \leftarrow left\_of(phone, telephone) \\
noisy(pencil) \leftarrow left\_of(phone, telephone) \\
noisy(pencil) \leftarrow left\_of(phone, pencil) \\
noisy(phone) \leftarrow noisy(telephone) \wedge noisy(pencil)
\end{array}
$$

Example Truths

In the interpretation given before, which of following are true?

$noisy(phone)$	true
$noisy(telephone)$	true
$noisy(pencil)$	false
$left\_of(phone, pencil)$	true
$left\_of(phone, telephone)$	false
$noisy(phone) \leftarrow left\_of(phone, telephone)$	true
$noisy(pencil) \leftarrow left\_of(phone, telephone)$	true
$noisy(pencil) \leftarrow left\_of(phone, pencil)$	false
$noisy(phone) \leftarrow noisy(telephone) \wedge noisy(pencil)$	true

Models and logical consequences (recall)

- A knowledge base, KB, is true in interpretation I if and only if every clause in KB is true in I.
- A **model** of a set of clauses is an interpretation in which all the clauses are true.
- If KB is a set of clauses and g is a conjunction of atoms, g is a **logical consequence** of KB, written $KB \models g$, if g is true in every model of KB.
- That is, $KB \models g$ if there is no interpretation in which KB is true and g is false.

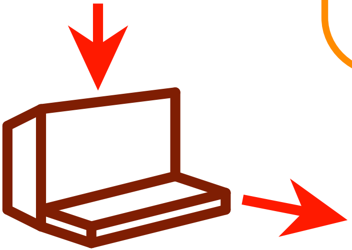
User's view of Semantics

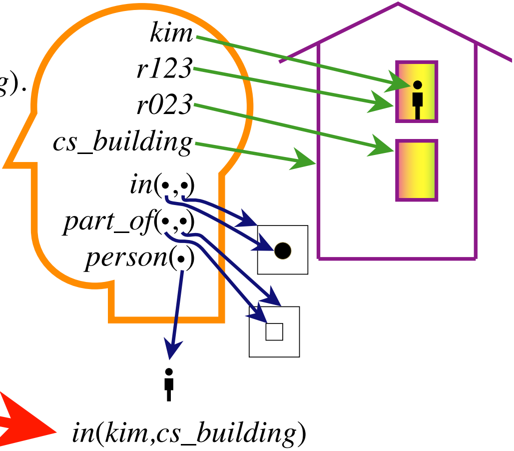
1. Choose a task domain: intended interpretation.
2. Associate constants with individuals you want to name.
3. For each relation you want to represent, associate a predicate symbol in the language.
4. Tell the system clauses that are true in the intended interpretation: axiomatizing the domain.
5. Ask questions about the intended interpretation.
6. If $KB \models g$, then g must be true in the intended interpretation.

Computer's view of semantics

- The computer doesn't have access to the intended interpretation.
- All it knows is the knowledge base.
- The computer can determine if a formula is a logical consequence of KB.
- If $KB \models g$ then g must be true in the intended interpretation.
- If $KB \not\models g$ then there is a model of KB in which g is false. This could be the intended interpretation.

$in(kim, r123).$
$part\_of(r123, cs\_building).$
$in(X, Y) \leftarrow$
$\quad part\_of(Z, Y) \land$
$\quad in(X, Z).$





Variables

- Variables are universally quantified in the scope of a clause.
- A variable assignment is a function from variables into the domain.
- Given an interpretation and a variable assignment,
  each term denotes an individual and
  each clause is either true or false.
- A clause containing variables is true in an interpretation if it is true for all variable assignments.

Queries and Answers

A **query** is a way to ask if a body is a logical consequence of the knowledge base:

$$?b_1 \wedge \cdots \wedge b_m.$$

An **answer** is either

- an instance of the query that is a logical consequence of the knowledge base KB, or
- **no** if no instance is a logical consequence of KB.

Example Queries

$$KB = \begin{cases} in(kim, r123). \\ part\_of(r123, cs\_building). \\ in(X, Y) \leftarrow part\_of(Z, Y) \wedge in(X, Z). \end{cases}$$

Query	Answer
$?part\_of(r123, B).$	

Example Queries

$$KB = \left\{ \begin{array}{l} in(kim, r123). \\ part\_of(r123, cs\_building). \\ in(X, Y) \leftarrow part\_of(Z, Y) \wedge in(X, Z). \end{array} \right.$$

Query	Answer
$?part\_of(r123, B).$	$part\_of(r123, cs\_building)$
$?part\_of(r023, cs\_building).$	

Example Queries

$$KB = \begin{cases} in(kim, r123). \\ part\_of(r123, cs\_building). \\ in(X, Y) \leftarrow part\_of(Z, Y) \wedge in(X, Z). \end{cases}$$

Query	Answer
$?part\_of(r123, B).$	$part\_of(r123, cs\_building)$
$?part\_of(r023, cs\_building).$	*no*
$?in(kim, r023).$	

Example Queries

$$KB = \begin{cases} in(kim, r123). \\ part\_of(r123, cs\_building). \\ in(X, Y) \leftarrow part\_of(Z, Y) \land in(X, Z). \end{cases}$$

Query	Answer
$?part\_of(r123, B).$	$part\_of(r123, cs\_building)$
$?part\_of(r023, cs\_building).$	no
$?in(kim, r023).$	no
$?in(kim, B).$	

Example Queries

$$KB = \begin{cases} in(kim, r123). \\ part\_of(r123, cs\_building). \\ in(X, Y) \leftarrow part\_of(Z, Y) \wedge in(X, Z). \end{cases}$$

Query	Answer
$?part\_of(r123, B).$	$part\_of(r123, cs\_building)$
$?part\_of(r023, cs\_building).$	no
$?in(kim, r023).$	no
$?in(kim, B).$	$in(kim, r123)$
	$in(kim, cs\_building)$

Logical Consequence

Atom g is a logical consequence of KB if and only if:

- g is a fact in KB, or
- there is a rule

  $$g \leftarrow b_1 \wedge \ldots \wedge b_k$$

  in KB such that each b_i is a logical consequence of KB.

Debugging false conclusions

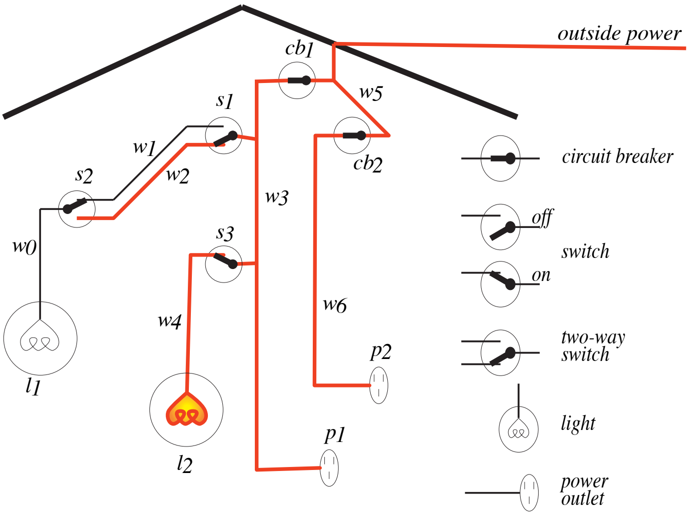
To debug answer g that is false in the intended interpretation:

- If g is a fact in KB, this fact is wrong.
- Otherwise, suppose g was proved using the rule:

  $$g \leftarrow b_1 \wedge \ldots \wedge b_k$$

  where each b_i is a logical consequence of KB.
  - If each b_i is true in the intended interpretation, this clause is false in the intended interpretation.
  - If some b_i is false in the intended interpretation, debug b_i.

Electrical Environment



Axiomatizing the Electrical Environment

% $light(L)$ is true if L is a light
$light(l_1)$. $light(l_2)$.
% $down(S)$ is true if switch S is down
$down(s_1)$. $up(s_2)$. $up(s_3)$.
% $ok(D)$ is true if D is not broken
$ok(l_1)$. $ok(l_2)$. $ok(cb_1)$. $ok(cb_2)$.

$?light(l_1)$. $\Longrightarrow$

Axiomatizing the Electrical Environment

% $light(L)$ is true if L is a light
$light(l_1)$. $light(l_2)$.
% $down(S)$ is true if switch S is down
$down(s_1)$. $up(s_2)$. $up(s_3)$.
% $ok(D)$ is true if D is not broken
$ok(l_1)$. $ok(l_2)$. $ok(cb_1)$. $ok(cb_2)$.

$?light(l_1)$. $\Longrightarrow$ yes
$?light(l_6)$. $\Longrightarrow$

Axiomatizing the Electrical Environment

% $light(L)$ is true if L is a light
$light(l_1)$. $light(l_2)$.
% $down(S)$ is true if switch S is down
$down(s_1)$. $up(s_2)$. $up(s_3)$.
% $ok(D)$ is true if D is not broken
$ok(l_1)$. $ok(l_2)$. $ok(cb_1)$. $ok(cb_2)$.

$?light(l_1)$. $\Longrightarrow$ *yes*
$?light(l_6)$. $\Longrightarrow$ *no*
$?up(X)$. $\Longrightarrow$

Axiomatizing the Electrical Environment

% $light(L)$ is true if L is a light
$light(l_1)$. $light(l_2)$.
% $down(S)$ is true if switch S is down
$down(s_1)$. $up(s_2)$. $up(s_3)$.
% $ok(D)$ is true if D is not broken
$ok(l_1)$. $ok(l_2)$. $ok(cb_1)$. $ok(cb_2)$.

$?light(l_1)$. $\Longrightarrow$ yes
$?light(l_6)$. $\Longrightarrow$ no
$?up(X)$. $\Longrightarrow$ $up(s_2)$, $up(s_3)$

$connected\_to(X, Y)$ is true if component X is connected to Y

$$connected\_to(w_0, w_1) \leftarrow up(s_2).$$
$$connected\_to(w_0, w_2) \leftarrow down(s_2).$$
$$connected\_to(w_1, w_3) \leftarrow up(s_1).$$
$$connected\_to(w_2, w_3) \leftarrow down(s_1).$$
$$connected\_to(w_4, w_3) \leftarrow up(s_3).$$
$$connected\_to(p_1, w_3).$$

$?connected\_to(w_0, W).$ $\Longrightarrow$

$connected\_to(X, Y)$ is true if component X is connected to Y

$$connected\_to(w_0, w_1) \leftarrow up(s_2).$$
$$connected\_to(w_0, w_2) \leftarrow down(s_2).$$
$$connected\_to(w_1, w_3) \leftarrow up(s_1).$$
$$connected\_to(w_2, w_3) \leftarrow down(s_1).$$
$$connected\_to(w_4, w_3) \leftarrow up(s_3).$$
$$connected\_to(p_1, w_3).$$

$?connected\_to(w_0, W).$ $\Longrightarrow$ $W = w_1$

$?connected\_to(w_1, W).$ $\Longrightarrow$

$connected\_to(X, Y)$ is true if component X is connected to Y

$$connected\_to(w_0, w_1) \leftarrow up(s_2).$$
$$connected\_to(w_0, w_2) \leftarrow down(s_2).$$
$$connected\_to(w_1, w_3) \leftarrow up(s_1).$$
$$connected\_to(w_2, w_3) \leftarrow down(s_1).$$
$$connected\_to(w_4, w_3) \leftarrow up(s_3).$$
$$connected\_to(p_1, w_3).$$

$?connected\_to(w_0, W).$ $\Longrightarrow$ $W = w_1$

$?connected\_to(w_1, W).$ $\Longrightarrow$ no

$?connected\_to(Y, w_3).$ $\Longrightarrow$

$connected\_to(X, Y)$ is true if component X is connected to Y

$$connected\_to(w_0, w_1) \leftarrow up(s_2).$$
$$connected\_to(w_0, w_2) \leftarrow down(s_2).$$
$$connected\_to(w_1, w_3) \leftarrow up(s_1).$$
$$connected\_to(w_2, w_3) \leftarrow down(s_1).$$
$$connected\_to(w_4, w_3) \leftarrow up(s_3).$$
$$connected\_to(p_1, w_3).$$

$?connected\_to(w_0, W).$ $\Longrightarrow$ $W = w_1$
$?connected\_to(w_1, W).$ $\Longrightarrow$ no
$?connected\_to(Y, w_3).$ $\Longrightarrow$ $Y = w_2,\ Y = w_4,\ Y = p_1$
$?connected\_to(X, W).$ $\Longrightarrow$

$connected\_to(X, Y)$ is true if component X is connected to Y

$$connected\_to(w_0, w_1) \leftarrow up(s_2).$$
$$connected\_to(w_0, w_2) \leftarrow down(s_2).$$
$$connected\_to(w_1, w_3) \leftarrow up(s_1).$$
$$connected\_to(w_2, w_3) \leftarrow down(s_1).$$
$$connected\_to(w_4, w_3) \leftarrow up(s_3).$$
$$connected\_to(p_1, w_3).$$

$?connected\_to(w_0, W). \implies W = w_1$

$?connected\_to(w_1, W). \implies no$

$?connected\_to(Y, w_3). \implies Y = w_2,\ Y = w_4,\ Y = p_1$

$?connected\_to(X, W). \implies X = w_0, W = w_1, \ldots$

% $lit(L)$ is true if the light L is lit

$$lit(L) \leftarrow light(L) \wedge ok(L) \wedge live(L).$$

% $live(C)$ is true if there is power coming into C

$$\begin{aligned} & live(Y) \leftarrow \\ & \qquad connected\_to(Y, Z) \wedge \\ & \qquad live(Z). \\ & live(outside). \end{aligned}$$

This is a recursive definition of *live*.

Recursion and Mathematical Induction

$$above(X, Y) \leftarrow on(X, Y).$$
$$above(X, Y) \leftarrow on(X, Z) \land above(Z, Y).$$

This can be seen as:

- Recursive definition of *above*: prove *above* in terms of a base case (*on*) or a simpler instance of itself; or
- Way to prove *above* by mathematical induction: the base case is when there are no blocks between X and Y, and if you can prove *above* when there are n blocks between them, you can prove it when there are $n + 1$ blocks.

Limitations

Suppose you had a database using the relation:

$$enrolled(S, C)$$

which is true when student S is enrolled in course C.
You can't define the relation:

$$empty\_course(C)$$

which is true when course C has no students enrolled in it.
This is because $empty\_course(C)$ doesn't logically follow from a set of *enrolled* relations. There are always models where someone is enrolled in a course!

KR & Logic: Expressivity // Uncertainty

- Expressivity
  - Which kind of statements can a logic express, with which kind of primitives?
    - Deontic Logics (Obligations/Permissions)
    - Epistemic Logics (Beliefs)
    - Spatial, Temporal, Spatio-temporal Logics (Topology, Directions, LTL,…)
    - Causality & Events
- Beyond 1/0: logics for reasoning about information incompleteness, uncertainty, vagueness
  - Information incompleteness: premixes are incomplete
    - Non Monotonic Logics (Negation as failure, Default, Autoepistemic)
  - Uncertainty as a first-class citizen: premixes & conclusions are uncertain
    - Multi-valued Logics, PSL (Probabilistic Soft Logic)
  - Vagueness: some concepts, e.g., "*tall*", are intrinsically vague
    - Fuzzy logics & fuzzy sets
    - Analogical reasoning & similarity

A formalization

S = {On(a,b), On(b,c), Green(a), ¬Green(c)}

all that is required

$\alpha = \exists x \exists y[\mathrm{Green}(x) \wedge \neg\mathrm{Green}(y) \wedge \mathrm{On}(x,y)]$

In FOL we had a consequence from two cases, here it's different: more hypothetical configurations, each one leading to a different inference

Claim: $S \models \alpha$

Proof:

Let $\Im$ be any interpretation such that $\Im \models S$.

Case 1: $\Im \models \mathrm{Green(b)}$.

$\therefore\ \Im \models \mathrm{Green(b)} \wedge \neg\mathrm{Green(c)} \wedge \mathrm{On(b,c)}$.

$\therefore\ \Im \models \alpha$

Case 2: $\Im \not\models \mathrm{Green(b)}$.

$\therefore\ \Im \models \neg\mathrm{Green(b)}$

$\therefore\ \Im \models \mathrm{Green(a)} \wedge \neg\mathrm{Green(b)} \wedge \mathrm{On(a,b)}$.

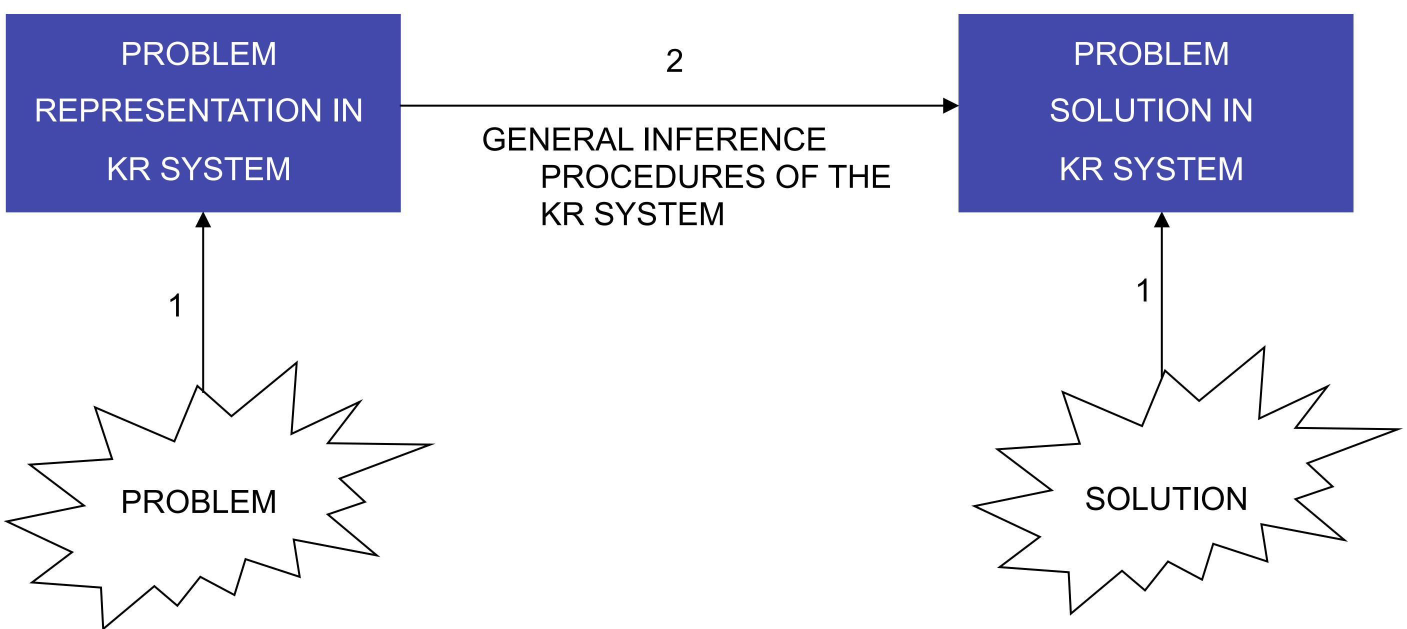
$\therefore\ \Im \models \alpha$

Either way, for any $\Im$, if $\Im \models S$ then $\Im \models \alpha$.

So $S \models \alpha$.   QED

Declarative Problem Solving & Logic-based KR

- Problem solving based on the definition of the problem ("what") and on the application of general strategies rather than on a set of instruction ("how")



Declarative Problem Solving & Logic-based KR

STRATIGRAPHY:

Define spatial relationships between strata

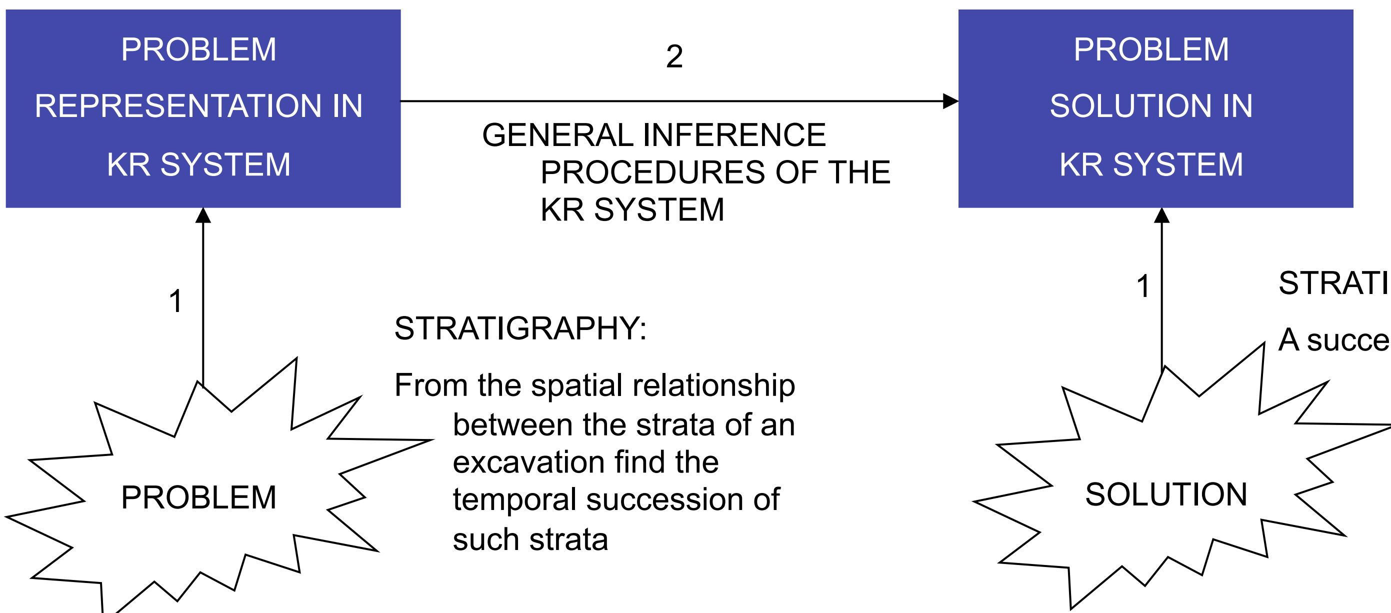
Define “what” is a correct temporal succession of strataa

Define the general correlations between spatial relationships holding among strata and their possible succession with respect to time

Give inputs: specific spatial relationships of an excavation

STRATIGRAPHY:

The temporal succession of the strata of the excavation



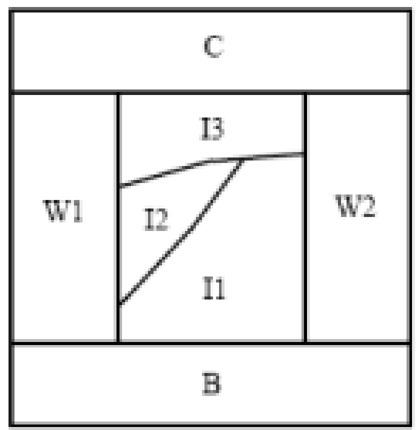
STRATIGRAPHY:

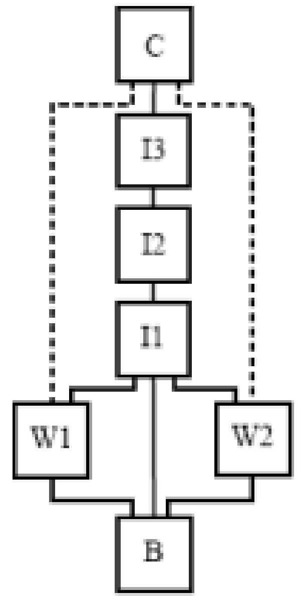
From the spatial relationship between the strata of an excavation find the temporal succession of such strata

STRATIGRAPHY:

A succession of the strata

Logic Programming Applied to Stratigraphy (Spatial to Temporal Mapping)





1

PROBLEM

STRATIGRAPHY:

From the spatial relationship between the strata of an excavation find the temporal succession of such strata

1

SOLUTION

STRATIGRAPHY:

A succession of the strata

Logic Programming Applied to Stratigraphy (Spatial to Temporal Mapping)

STRATIGRAPHY:

Define spatial relationships between strata

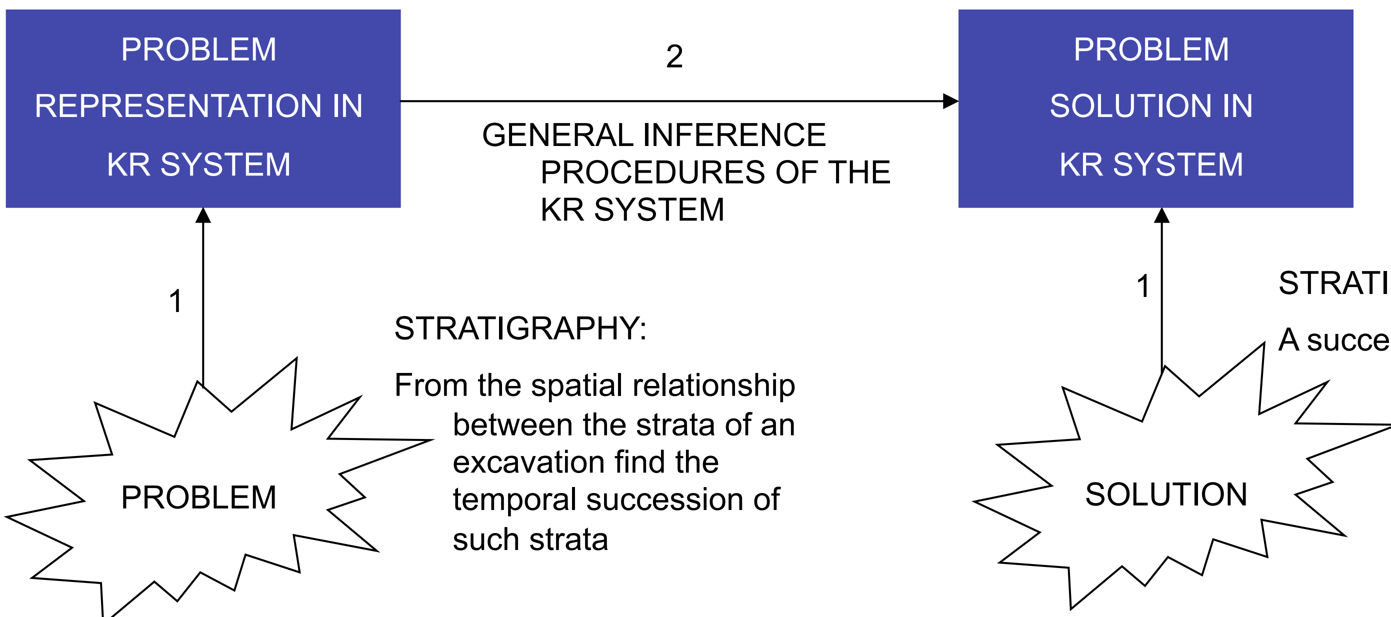
Define “what” is a correct temporal succession of strataa

Define the general correlations between spatial relationships holding among strata and their possible succession with respect to time

Give inputs: specific spatial relationships of an excavation

STRATIGRAPHY:

The temporal succession of the strata of the excavation



STRATIGRAPHY:

From the spatial relationship between the strata of an excavation find the temporal succession of such strata

STRATIGRAPHY:

A succession of the strata

Logic Programming Applied to Stratigraphy (Spatial to Temporal Mapping)

BACKGROUND KNOWLEDGE
ABOUT STRATIGRAPHY

STRATIGRAPHY:

Define spatial relationships between strata

Define "what" is a correct temporal succession of strataa

Define the general correlations between spatial relationships holding among strata and their possible succession with respect to time

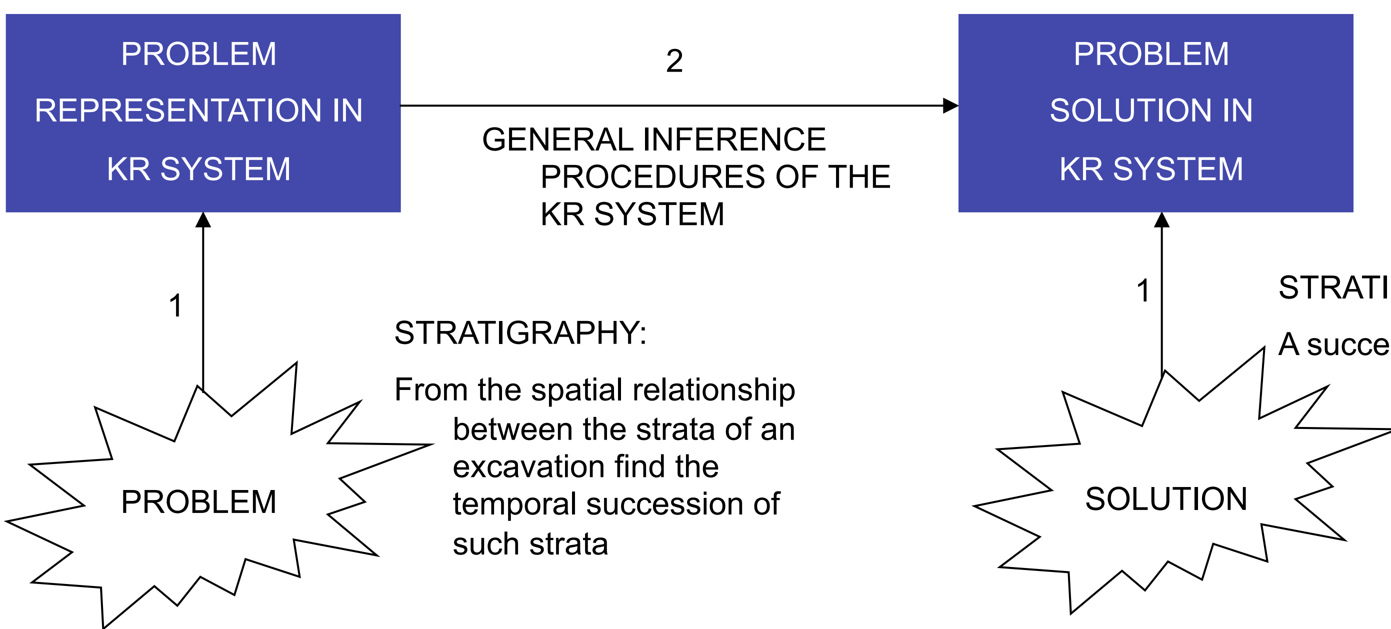
Give inputs: specific spatial relationships of an excavation

PROBLEM INPUTS

STRATIGRAPHY:

The temporal succession of the strata of the excavation

PROBLEM OUTPUT



STRATIGRAPHY:

From the spatial relationship between the strata of an excavation find the temporal succession of such strata

STRATIGRAPHY:

A succession of the strata

Logic Programming Applied to Stratigraphy (Spatial to Temporal Mapping)

Incomplete Knowledge

STRATIGRAPHY:

Define spatial relationships between strata

Define “what” is a correct temporal succession of strataa

Define the general correlations between spatial relationships holding among strata and their possible succession with respect to time

Give inputs: specific spatial relationships of an excavation

STRATIGRAPHY:

The temporal succession of the strata of the excavation

PROBLEM REPRESENTATION IN KR SYSTEM

2

GENERAL INFERENCE PROCEDURES OF THE KR SYSTEM

PROBLEM SOLUTION IN KR SYSTEM

1

STRATIGRAPHY:

From the spatial relationship between the strata of an excavation find the temporal succession of such strata

PROBLEM

1

STRATIGRAPHY:

The possible successions of the strata

SOLUTION

Primitive spatial relationships

```
-cover(Y,X) :- cover(X,Y).
coveredBy(X,Y) :- cover(Y,X).
cover(X,Y) :- coveredBy(Y,X).
```

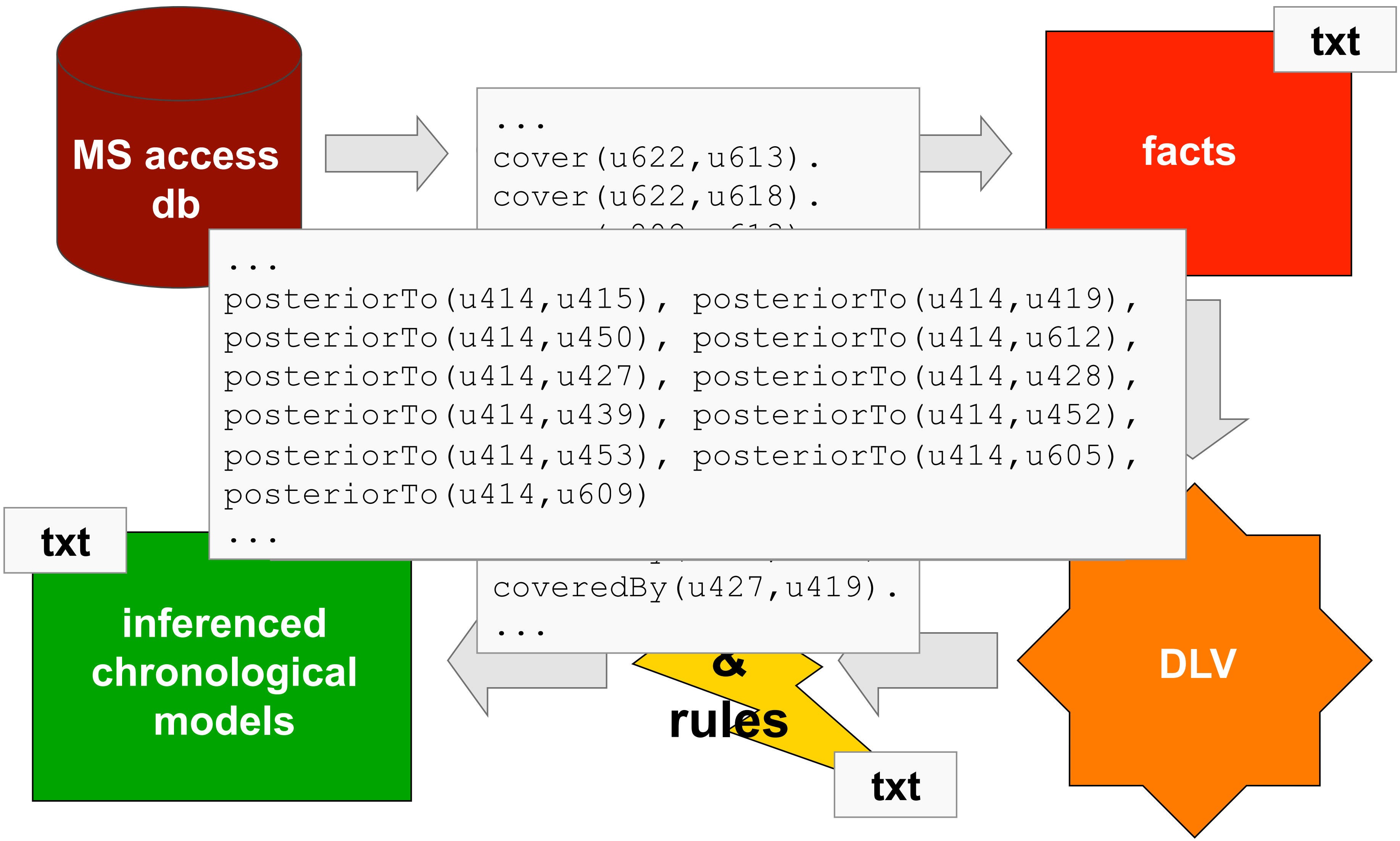
Primitive temporal relationships

```
:- dirPostTo(X,X).
-dirPostTo(X,Y) :- dirPostTo(Y,X).

:- dirAntTo(X,X).
```

Mixed axioms & multiple model generation

```
posteriorTo(Y,W) :- leanOn(X,Y),leanOn(X,Z),cover(Y,Z),cover(X,W),leanOn(W,Z).

posteriorTo(X,W) :- leanOn(X,Y),leanOn(Z,W),cover(X,Z),cover(Y,W).
posteriorTo(Y,Z) :- leanOn(X,Y),leanOn(Z,W),cover(X,Z),cover(Y,W).

posteriorTo(Z,Y) :- contemporary(X,Y),posteriorTo(Z,X).
posteriorTo(Z,X) :- contemporary(X,Y),posteriorTo(Z,Y).

dirPostTo(Z,Y) :- equalTo(X,Y),cover(Z,X).
dirPostTo(Z,X) :- equalTo(X,Y),cover(Z,Y).

-contemporary(Z,Y) :- equalTo(X,Y),posteriorTo(X,Z).
-contemporary(Z,X) :- equalTo(X,Y),posteriorTo(Y,Z).

contemporary(X,Y) v posteriorTo(X,Y) v posteriorTo(Y,X) :- us(X),us(Y),not posteriorTo(X,Y),not -posterior(X,Y).
```

```
dirPostTo(X,Y) :- fill(X,Y).
dirPostTo(X,Y) :- leanOn(X,Y).

contemporary(X,Y) :- attachTo(X,Y).
contemporary(X,Y) :- equalTo(X,Y).
-contemporary(X,Y) :- -equalTo(X,Y).
```


MS access db
...
cover(u622,u613).
cover(u622,u618).
facts
txt
...
posteriorTo(u414,u415), posteriorTo(u414,u419),
posteriorTo(u414,u450), posteriorTo(u414,u612),
posteriorTo(u414,u427), posteriorTo(u414,u428),
posteriorTo(u414,u439), posteriorTo(u414,u452),
posteriorTo(u414,u453), posteriorTo(u414,u605),
posteriorTo(u414,u609)
...
txt
inferenced chronological models
coveredBy(u427,u419).
...
&
rules
txt
DLV

Lack of knowledge generates multiple models

DLV [build BEN/Oct 11 2007   gcc 3.4.5 (mingw special)]

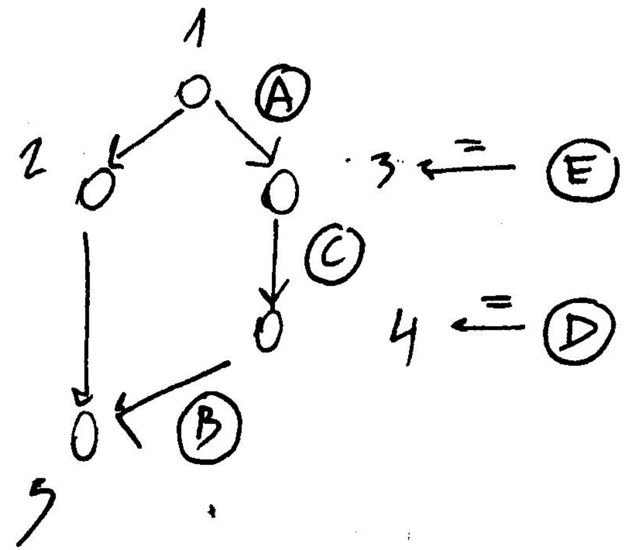
(A) {posteriorTo(us1,us2), posteriorTo(us1,us3), posteriorTo(us2,us5),
posteriorTo(us3,us5), posteriorTo(us3,us4), posteriorTo(us4,us5),
posteriorTo(us2,us3), posteriorTo(us2,us4)}

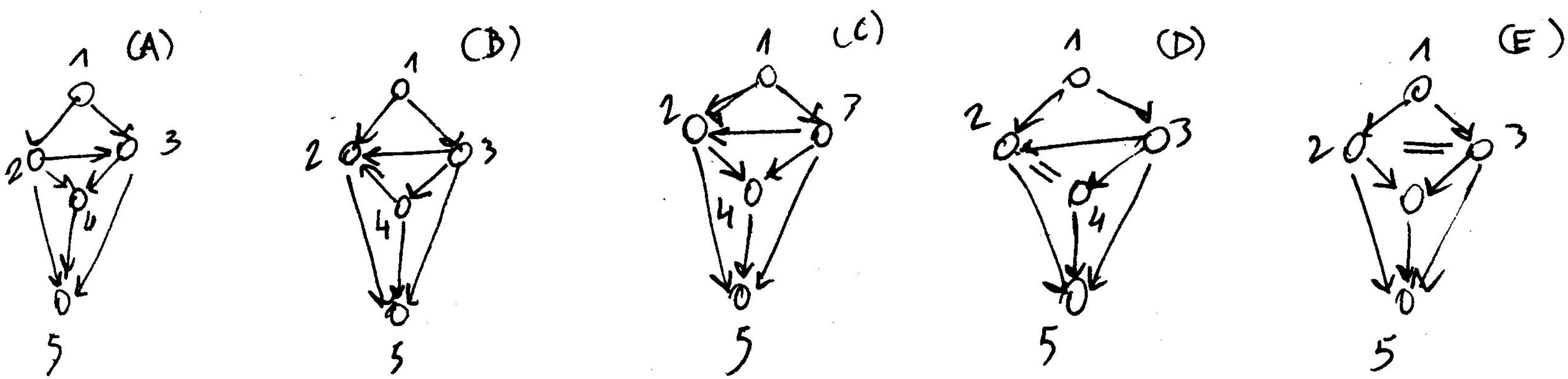
(B) {posteriorTo(us1,us2), posteriorTo(us1,us3), posteriorTo(us2,us5),
posteriorTo(us3,us5), posteriorTo(us3,us4), posteriorTo(us4,us5),
posteriorTo(us3,us2), posteriorTo(us4,us2)}

(C) {posteriorTo(us1,us2), posteriorTo(us1,us3), posteriorTo(us2,us5),
posteriorTo(us3,us5), posteriorTo(us3,us4), posteriorTo(us4,us5),
posteriorTo(us3,us2), posteriorTo(us2,us4)}

(D) {posteriorTo(us1,us2), posteriorTo(us1,us3), posteriorTo(us2,us5),
posteriorTo(us3,us5), posteriorTo(us3,us4), posteriorTo(us4,us5),
posteriorTo(us3,us2), contemporary(us4,us2), contemporary(us2,us4)}

(E) {posteriorTo(us1,us2), posteriorTo(us1,us3), posteriorTo(us2,us5),
posteriorTo(us3,us5), posteriorTo(us3,us4), posteriorTo(us4,us5),
contemporary(us3,us2), posteriorTo(us2,us4), contemporary(us2,us3)}





Expressivity of Logic Programs and Answer Set Programming (ASP)

A: an **atom**, i.e., $A(t_1, \ldots, t_h)$

L : a **literal**, i.e., A or –A

"-" is the true negation, different from "not"

$$B_1 \leftarrow A_1 \wedge \ldots \wedge A_m$$

Horn clause

A clause with *at most* one positive literal in the head

Expressivity of Logic Programs and Answer Set Programming (ASP)

disjunctive extended LPs

L_0 **or**... **or** L_k:- L_{k+1} , ... , L_m , not L_{m+1} , ... , not L_n	rule : head $\leftarrow$ body
L_0 :-	fact
:- L_0 , ... , L_m , not L_{m+1} , ... , not L_n	constraint ($\perp$)

A_k:- A_{k+1} , ... , A_m	definite LPs	ASP^{-not}
A_0 :- A_1 , ... , A_m , not A_{m+1} , ... , not A_n	normal LPs	ASP
L_0 :- L_1 , ... , L_m , not L_{m+1} , ... , not L_n	extended LPs	$ASP^{\neg}$
A_0 **or**... **or** A_k:- A_{k+1} , ... , A_m , not A_{m+1} , ... , not A_n	disjunctive LPs	ASP^{or}
L_0 **or**... **or** L_k:- L_{k+1} , ... , L_m , not L_{m+1} , ... , not L_n		ASP^{*}=$ASP^{\neg,\perp,or}$

Expressivity of Logic Programs and Answer Set Programming (ASP)

datalog = no simboli di funzione*

	disjunctive extended LPs
L_0 **or**... **or** L_k:- L_{k+1} , ... , L_m , not L_{m+1} , ... , not L_n	rule : head $\leftarrow$ body
L_0 :-	fact
:- L_0 , ... , L_m , not L_{m+1} , ... , not L_n	constraint ($\perp$)

A_k:- A_{k+1} , ... , A_m	definite LPs	ASP^{-not}
A_0 :- A_1 , ... , A_m , not A_{m+1} , ... , not A_n	normal LPs	ASP
L_0 :- L_1 , ... , L_m , not L_{m+1} , ... , not L_n	extended LPs	$ASP^{\neg}$
A_0 **or**... **or** A_k:- A_{k+1} , ... , A_m , not A_{m+1} , ... , not A_n	disjunctive LPs	ASP^{or}
L_0 **or**... **or** L_k:- L_{k+1} , ... , L_m , not L_{m+1} , ... , not L_n		$ASP^{*}=ASP^{\neg,\perp,or}$