Well-Founded Semantics for Deductive Object-Oriented Database Languages

Wolfgang May

Bertram Ludäscher

Georg Lausen

Institut für Informatik

Universität Freiburg

Germany

December 11, 1997

- Well-Founded Semantics
- Alternating-Fixpoint Characterization
- Deductive Object Oriented Languages
- Functional Methods ?
- Inheritance ?

The Well-Founded Semantics

- A. Van Gelder, K. Ross, and J. Schlipf:
  Unfounded Sets and Well-Founded Semantics for General Logic Programs. In *Proc. ACM Symposium on Principles of Database Systems (PODS)*, pages 221–230, 1988.
- Generally accepted as a sceptical "well-behaved" semantics for logic programs with negation.
- Assigns a unique, three-valued model to every program. *undefined* is assigned to atoms which depend negatively on themselves and for which no independent "well-founded" derivation exists.
- several logic programming languages (e.g. XSB-Prolog) and relational database systems now support WFS.

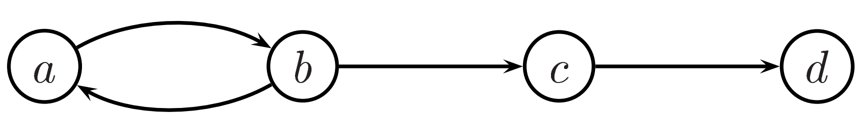
Well-Founded Semantics in DOOD languages?

- *dood* systems currently limited to inflationary or stratified semantics.

(Why) do we need WFS ?

- Stratification:
- relational: notion of stratification is based on explicit dependencies between relation symbols.
- OO: dependencies are conceptually more involved: value inheritance, a dynamic class hierarchy, and higher-order features like variables at method or class positions.
  $\Rightarrow$ In general non-stratified programs.
- stratified negation is less expressive than well-founded negation, certain concepts cannot be expressed in stratified semantics due to cyclic negative dependencies: deep equality, argumentation frameworks.
- Example: Win-Move Game.
  A set of positions and a set of moves between them, two players moving alternately; a player who cannot move loses.

  win(X) ←move(X,Y), ¬ win(Y).



Object-Oriented Model

- is-a atoms: o:c
  relational encoding: isa(o,c)
- subclass atoms: c::d
  relational encoding: subcl(o,c)
- Method applications to objects:
  o[m→v] (scalar)
  o[m↠v] (multivalued)
  analogous with arguments: o[m@(x_1,...,x_n)→v].
  inheritable:
  o[m•→v]
  o[m•↠v]

  relational encoding:
  method_appl_sc(o,m,v), method_appl_mvd(o,m,v)
- path expressions: $o.m \;\equiv\; o'$ s.t. $o[m{\rightarrow}o']$
- Variables: Capital letters;
- Inheritance
- Transitivity of subclass hierarchy

Stratified Semantics

- Relational: based on the dependency graph
- p depends positively/negatively on q if there is a rule with p occurring in the head and q occurring positively/negatively in the body.
- P is *stratified* if it does not contain a cyclic negative dependency.

Stratified Semantics in DOOD languages?

Dependency graph in *dood* languages:

distinguished positions:

o[m →v]   o:c   c::d

x *depends positively (negatively)* on a symbol y if there is a rule r s.t. x occurs at the distinguished position of the head of r and y occurs in a positive (negative) literal in the body.

Practical solution if

- no inheritance or
- static class hierarchy and membership,
- no variables at method name or class positions.

Inherently non-stratifiable Constructs

- Non-monotonic inheritance:
  c[m•→v] an inheritable scalar method of the class *c*.

  $$\mathsf{O[m{\rightarrow}v]} \;\leftarrow\; \mathsf{c[m{\bullet}{\rightarrow}v],\ O{:}c,}$$
  $$\mathsf{not}\ \exists\ \mathsf{W{:}\ (O[m{\rightarrow}W]} \wedge \mathsf{W} \neq \mathsf{v).}$$

  application of an inheritable method to an object depends negatively on itself ⇒ not stratifiable.

- Variables at method or class positions:

  $$\mathsf{o[M{\rightarrow}v]} \quad , \quad \mathsf{o{:}C}$$

  potentially replaced by an arbitrary symbol
  ⇒ very "dense" dependency graph.

Alternating-Fixpoint Characterization

Given a Herbrand interpretation $\mathbf{J}$ and a logic program P, $T_P^{\mathbf{J}}$, mapping interpretations to interpretations is defined as

$$\begin{aligned} T_P^{\mathbf{J}}(\mathbf{I}) := \{H \mid \ & (H \leftarrow B_1, \ldots, B_n, \neg C_1, \ldots, \neg C_m) \in grd(P) \\ & \text{and } B_i \in \mathbf{I} \text{ for all } i = 1, \ldots, n \\ & \text{and } C_j \notin \mathbf{J} \text{ for all } j = 1, \ldots, m \ \} \end{aligned}$$

- $T_P^{\mathbf{J}}$ is monotone (in $\mathbf{I}$).
- $\Gamma_P(\mathbf{J}) := \mathrm{lfp}(T_P^{\mathbf{J}})$ is antimonotone (in $\mathbf{J}$)
  $(\mathbf{J}_1 \subseteq \mathbf{J}_2 \rightsquigarrow \Gamma_P(\mathbf{J}_2) \subseteq \Gamma_P(\mathbf{J}_1))$
- Γ_P^2 $(:= \Gamma_P \circ \Gamma_P)$ is monotone.
- $\emptyset, \Gamma_P^2, \Gamma_P^4, \ldots$ is a monotonically growing sequence of underestimates of the true atoms, converging against $\mathrm{lfp}(\Gamma_P^2)$,
- $\Gamma_P^1, \Gamma_P^3, \ldots$ is a monotonically decreasing sequence of overestimates, converging against $\mathrm{gfp}(\Gamma_P^2)$.

Theorem 1 *For every ground atom A,*

$$\mathbf{W}(P)(A) = \begin{cases} \textit{true} & \textit{if } A \in \mathrm{lfp}(\Gamma_P^2), \\ \textit{false} & \textit{if } A \notin \mathrm{gfp}(\Gamma_P^2), \\ \textit{undef} & \textit{if } A \in \mathrm{gfp}(\Gamma_P^2) \setminus \mathrm{lfp}(\Gamma_P^2). \end{cases}$$

Computing WFS via States.

Compute $\emptyset, \Gamma_P^1, \Gamma_P^2, \ldots$ by a logic program:

- an additional argument position for IDB-relations: $r(x_1, \ldots, x_n) \rightsquigarrow r(s, x_1, \ldots, x_n)$
- set s to $S{+}1$ for all positive literals (including the head literal),
- set s to S for negative literals.

$$T_P^{\mathbf{J}}(\mathbf{I}) := \{H \mid \ (H \leftarrow B_1, \ldots, B_n, \neg C_1, \ldots, \neg C_m) \in grd(P) \text{ and } B_i \in \mathbf{I} \text{ for all } i = 1, \ldots, n \text{ and } C_j \notin \mathbf{J} \text{ for all } j = 1, \ldots, m\ \}$$

- *state variable* S is restricted by state(S).

Example:

win(X) ←move(X,Y), ¬ win(Y).

win(S+1, X) ← move(X,Y), ¬ win(S, Y), state(S).
state(0).
state(S+1) ← state(S).

Computing WFS via States.

Compute $\emptyset, \Gamma_P^1, \Gamma_P^2, \ldots$ by a logic program:

- an additional argument position for IDB-relations: $r(x_1, \ldots, x_n) \leadsto r(s, x_1, \ldots, x_n)$
- set s to $S+1$ for all positive literals (including the head literal),
- set s to S for negative literals.

$$T_P^{\mathbf{J}}(\mathbf{I}) := \{H \mid \ (H \leftarrow B_1, \ldots, B_n, \neg C_1, \ldots, \neg C_m) \in grd(P) \text{ and } B_i \in \mathbf{I} \text{ for all } i = 1, \ldots, n \text{ and } C_j \notin \mathbf{J} \text{ for all } j = 1, \ldots, m\ \}$$

- negative dependencies only to the predecessor state and to EDB relations,

⇒ no cyclic negative dependencies between state-ground atoms,

⇒ state-stratified / effectively stratified.

- WFS can be computed also by systems which do not originally provide a WFS:
- By successively instantiating S with $0, 1, 2, \ldots$, precisely the AFP computation is obtained.
- Given a finite database, the computation finally becomes stationary or 2-periodic.

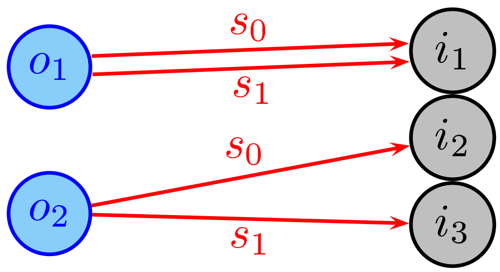
Representation of States

Relational Model:

Reification: $r(x_1, \ldots, x_n) \rightsquigarrow r(s, x_1, \ldots, x_n)$.

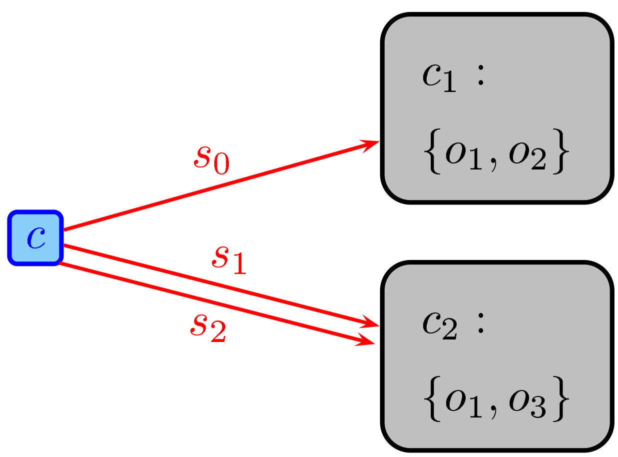
Object-Oriented Model:

Dynamic objects: For an abstract object o, a state s is a method, giving the instance of o corresponding to state s.



o.s[m → v]

Dynamic classes: For an abstract class c, a state s is a method, giving the instance c_s of the class c in this state.



o.s:c.s

- "dynamic objects" and "dynamic classes": states appear as methods, thus state variables appear as variables at method positions.

Alternating-Fixpoint Characterization

- associating states to atoms:

$$\begin{aligned} \mathsf{o}[\mathsf{m} \to \mathsf{v}][\![s]\!] &:= \mathsf{o}.\mathsf{s}[\mathsf{m} \to \mathsf{v}]\ , \\ \mathsf{o}{:}\mathsf{c}[\![s]\!] &:= \mathsf{o}.\mathsf{s}{:}\mathsf{c}.\mathsf{s} \\ \mathsf{c}{::}\mathsf{d}[\![s]\!] &:= \mathsf{c}.\mathsf{s}{::}\mathsf{d}.\mathsf{s} \end{aligned}$$

- AFP Transformation: For every rule $\mathsf{h} \leftarrow \mathsf{b}$,
  - EDB literals (occurring only in the body) remain unchanged,
  - every positive IDB literal ℓ is replaced by $\ell[\![S{+}1]\!]$,
  - every negative IDB literal $\neg\ell$ (which can occur only in the body) is replaced by $\neg\ell[\![S]\!]$, and
  - the body is extended with the literal $\mathsf{S{+}1{:}state}$.

Additionally, there are rules $\mathsf{0{:}state}$ and $\mathsf{S{+}1{:}state} \leftarrow \mathsf{S{:}state}$.

- negative dependencies only to the predecessor state and to EDB atoms without state associations,
- the state sequence provides a *local* stratification (*state stratification*) $\leadsto$ unique perfect model,
- by successively instantiating S with $0, 1, 2, \ldots,$ the AFP computation is obtained.

Alternating-Fixpoint Characterization

- The program must now be evaluated accordingly, i.e., one state after another.
- check if a deductive fixpoint is reached and then start the next deductive fixpoint,
- check if the state sequence becomes stationary or 2-periodic,
- yields a finite structure $\mathbf{A}_P$.

W.l.o.g., the last state which has been computed has an even index s_0. For every s s.t. $\mathbf{A}_P \models (s : \mathsf{state})$, let

$$\begin{aligned} \mathbf{A}_P^{[s]} := \quad & \{a \mid \mathbf{A}_P \models a[\![s]\!],\ a \text{ an IDB atom}\} \cup \\ & \{a \mid \mathbf{A}_P \models a,\ a \text{ an EDB atom}\} \end{aligned}$$

("snapshot" at state s)

Theorem 2 *The well-founded model $\mathbf{W}_P$ is given as*

$$\mathbf{W}_P(a) = \begin{cases} \mathit{true} & \Leftrightarrow \ \mathbf{A}_P^{[s_0]} \models a \\ \mathit{undef} & \Leftrightarrow \ \mathbf{A}_P^{[s_0]} \models \neg a \ \mathit{and}\ \mathbf{A}_P^{[s_0-1]} \models a \\ \mathit{false} & \Leftrightarrow \ \mathbf{A}_P^{[s_0-1]} \models \neg a\ . \end{cases}$$

Functional Methods

- overestimates (odd s): there can be $v_1 \neq v_2$ s.t. $\mathbf{A}_P \models [m \rightarrow v_1][\![s]\!]$ and $\mathbf{A}_P \models [m \rightarrow v_2][\![s]\!]$
- functionality requirement violated in overestimates.
- $\mathbf{W}(o[m \rightarrow v]) = \mathit{undef}$ for *several* v's
- functionality requirement violated for undefined atoms.

Example: John is either married to Jane or to Mary:

P := { john[spouse→mary] ← not john[spouse→jane].
john[spouse→jane] ← not john[spouse→mary].
O[married→true] ← O[spouse→X] . }

john[spouse→mary]$[\![S{+}1]\!]$ ← not john[spouse→jane]$[\![S]\!]$, S+1:state.
john[spouse→jane]$[\![S{+}1]\!]$ ← not john[spouse→mary]$[\![S]\!]$, S+1:state.
O[married→true]$[\![S{+}1]\!]$ ← O[spouse→X]$[\![S{+}1]\!]$, S+1:state.
0:state.
S+1 : state ← S : state.

$\mathbf{A}_P^{[0]} = \emptyset$,
$\mathbf{A}_P^{[1]} = \{\text{john[spouse}\rightarrow\{\text{jane,mary}\}], \text{john[married}\rightarrow\text{true]}\}$ and
$\mathbf{A}_P^{[2]} = \emptyset$, periodic for $s_0 = 2$.

$\mathbf{W}$(john[spouse→jane]) = $\mathbf{W}$(john[spouse→mary]) =
$\mathbf{W}$(john[married→true]) = undef .

Inheritance

Semantics of inheritable methods:

$$\text{O[M}\rightarrow\text{V]} \leftarrow \text{C[M}\bullet\!\!\rightarrow\text{V], O:C,}$$
$$\text{not } \exists \text{ W: (O[M}\rightarrow\text{W]} \wedge \text{W} \neq \text{V).}$$
$$\text{D[M}\bullet\!\!\rightarrow\text{V]} \leftarrow \text{C[M}\bullet\!\!\rightarrow\text{V], D::C,}$$
$$\text{not } \exists \text{ W: (D[M}\rightarrow\text{W]} \wedge \text{W} \neq \text{V).}$$

(only to *direct* subclasses)

→ uses implicit negation

→ hard-code inheritance in P_{AFP}:

$$\text{O[M}\rightarrow\text{V]}[\![S{+}1]\!] \leftarrow \text{C[M}\bullet\!\!\rightarrow\text{V]}[\![S{+}1]\!]\text{, (O:C)}[\![S{+}1]\!]\text{,}$$
$$\neg\exists \text{ W: (O[M}\rightarrow\text{W]}[\![S]\!] \wedge \text{W} \neq \text{V),}$$
$$\neg\exists \text{ D: (O:D)}[\![S{+}1]\!] \wedge \text{(D::C)}[\![S{+}1]\!].$$

Analogous for classes:

$$\text{C'[M}\bullet\!\!\rightarrow\text{V]}[\![S{+}1]\!] \leftarrow \text{C[M}\bullet\!\!\rightarrow\text{V]}[\![S{+}1]\!]\text{, (C'::C)}[\![S{+}1]\!]\text{,}$$
$$\neg\exists \text{ W: (C'[M}\bullet\!\!\rightarrow\text{W]}[\![S]\!] \wedge \text{W} \neq \text{V),}$$
$$\neg\exists \text{ D: (C'::D)}[\![S{+}1]\!] \wedge \text{(D::C)}[\![S{+}1]\!].$$

→ and *disable* built-in inheritance (uses implicit negation).

Analogous for multivalued methods:
The *whole set* of values is inherited *iff* otherwise it would be undefined.

Inheritance for Multivalued Methods

The *whole set* of values is inherited *iff* otherwise it would be undefined:

$$
\begin{array}{l}
\mathsf{O}[\mathsf{M}{\rightarrow\!\!\!\rightarrow}\mathsf{V}][\![S{+}1]\!] \leftarrow \mathsf{C}[\mathsf{M}{\bullet\!\!\rightarrow\!\!\!\rightarrow}\mathsf{V}][\![S{+}1]\!],\ (\mathsf{O}{:}\mathsf{C})[\![S{+}1]\!],\\
\qquad ((\neg\exists\ \mathsf{W}{:}\ \mathsf{O}[\mathsf{M}{\rightarrow\!\!\!\rightarrow}\mathsf{W}][\![S]\!]) \vee\\
\qquad\quad \forall\ \mathsf{W}{:}\ \mathsf{O}[\mathsf{M}{\rightarrow\!\!\!\rightarrow}\mathsf{W}][\![S]\!] \leftrightarrow \mathsf{C}[\mathsf{M}\ {\bullet\!\!\rightarrow\!\!\!\rightarrow}\mathsf{W}][\![S{-}1]\!]),\\
\qquad \neg\exists\ \mathsf{D}{:}\ \ (\mathsf{O}{:}\mathsf{D})[\![S{+}1]\!] \wedge (\mathsf{D}{::}\mathsf{C})[\![S{+}1]\!].
\end{array}
$$

Features:

- Negation
- Inheritance

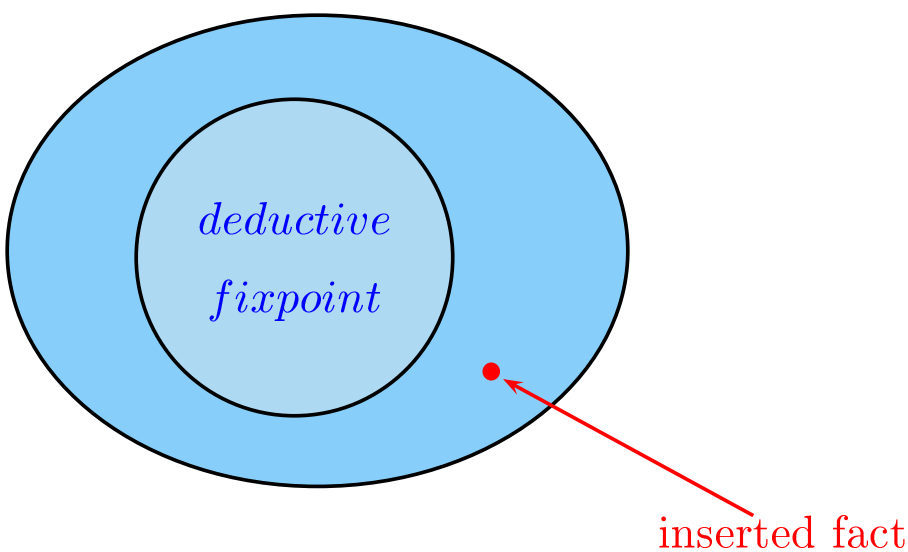
although *not* provided by the original system

Requirements

- State-Stratified Evaluation
- Fixpoint/2-Periodicity Detection

F-Logic

- Syntax: See Slide 5.
- Semantics (also implemented in the FLORID Prototype):
  - Inflationary semantics,
  - user-defined stratification (fixed number of predefined strata),
  - Trigger mechanism: Insert atoms into the database after reaching a deductive fixpoint (used for nonmonotonic inheritance).



Programming Explicit States in F-Logic

0:state.

0:even.

state[ready•→true].

state[running•→false].

S:state ← T[running→true], T.ready[ ], S = T + 1.

S:even ← S:state, S = T + 1, T:odd.

S:odd ← S:state, S = T + 1, T:even.

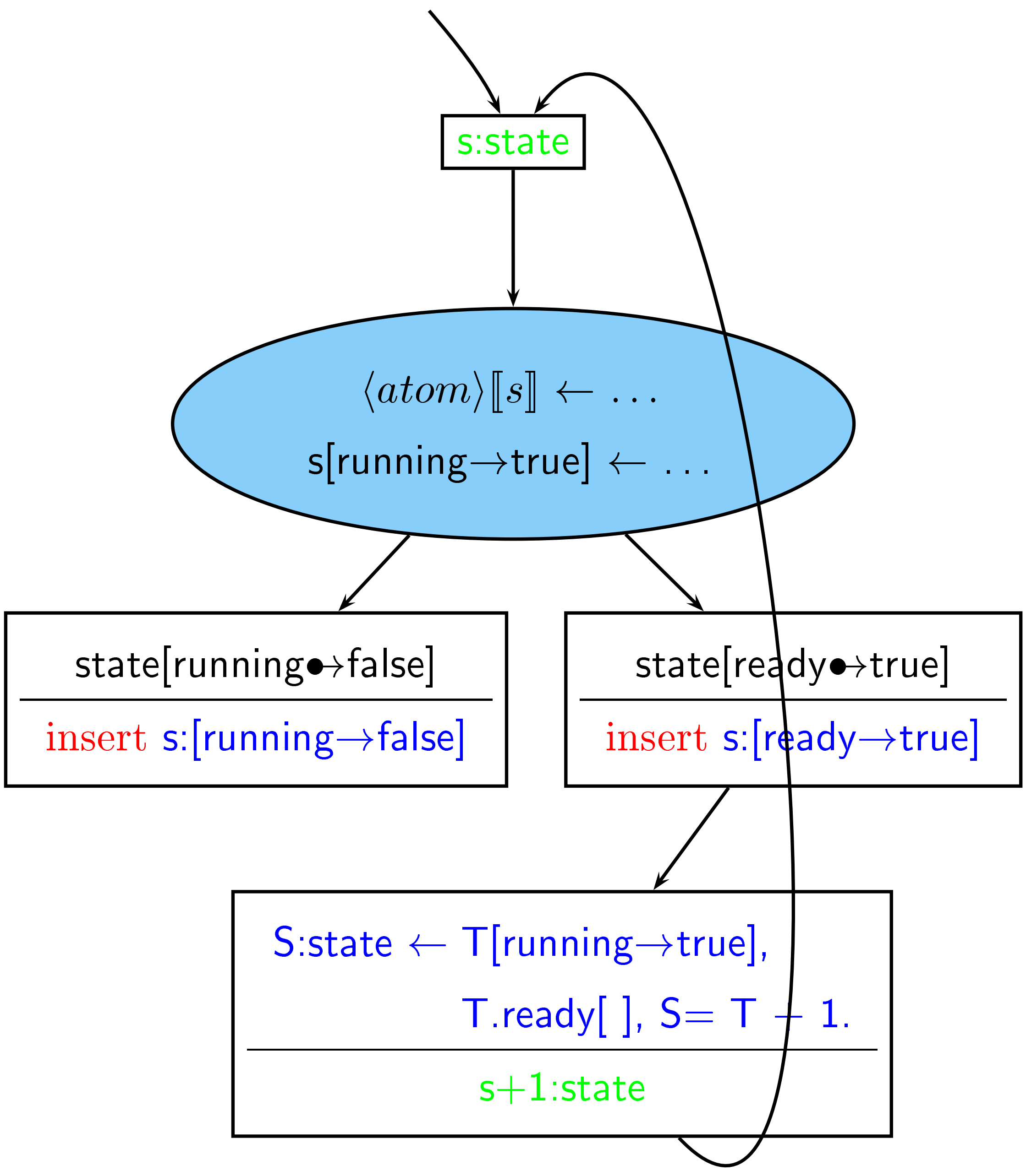
0[running→true].

S[running→true] ← S:odd.

S[running→true] ← not $\langle atom \rangle [\![T]\!]$, T:even,
$\langle atom \rangle [\![S]\!]$, S = T + 2.

S:final ← S[running→false].

The State Sequence



F-Logic

- Uniform Domain: Every element of the domain can occur as an object, a class, or a method (also polymorphous; scalar, multivalued, inheritable/ non-inheritable, with different numbers of arguments).
- create different *method objects* for scalar and multivalued and non-inheritable/inheritable method applications:
  - m.sc := m' s.t. m[sc →m'], scalar non-inheritable,
  - m.mvd := m' s.t. m[mvd →m'], scalar non-inheritable,
  - m.sci := m'' s.t. m[sc →m'] and m'[inh →m'']; scalar inheritable,
  - m.mvi := m'' s.t. m[mvd →m'] and m'[inh →m'']; multivalued inheritable.
- Then apply these *method objects* to objects: o[(m.sci)↠v]

Program Transformation

Replace method applications accordingly:

- replace all occurrences of scalar method applications $m{\rightarrow}$ by $m.\mathsf{sc} \twoheadrightarrow$:

$$\mathsf{O[M@(X_1,\ldots,X_n)} \rightarrow \mathsf{V]} \mapsto \mathsf{O[M.sc@(X_1,\ldots,X_n)} \twoheadrightarrow \mathsf{V]} \;,$$

- replace all occurrences of multivalued method applications $m{\twoheadrightarrow}$ by $m.\mathsf{mvd} \twoheadrightarrow$:

$$\mathsf{O[M@(X_1,\ldots,X_n)} \twoheadrightarrow \mathsf{V]} \mapsto \mathsf{O[M.mvd@(X_1,\ldots,X_n)} \twoheadrightarrow \mathsf{V]} \;,$$

- replace all occurrences of inheritable scalar method applications $m \bullet\!\!\rightarrow$ by $m.\mathsf{sci} \twoheadrightarrow$:

$$\mathsf{O[M@(X_1,\ldots,X_n)} \bullet\!\!\rightarrow \mathsf{V]} \mapsto \mathsf{O[M.sci@(X_1,\ldots,X_n)} \twoheadrightarrow \mathsf{V]} \;.$$

- replace all occurrences of inheritable multivalued method applications $m \bullet\!\!\twoheadrightarrow$ by $m.\mathsf{mvi} \twoheadrightarrow$:

$$\mathsf{O[M@(X_1,\ldots,X_n)} \bullet\!\!\twoheadrightarrow \mathsf{V]} \mapsto \mathsf{O[M.mvi@(X_1,\ldots,X_n)} \twoheadrightarrow \mathsf{V]} \;.$$

Program Transformation

Add rules which implement well-founded inheritance:

$$\text{O[M.sc}\twoheadrightarrow\text{V]}[\![S{+}1]\!] \leftarrow \text{C[M.sci}\twoheadrightarrow\text{V]}[\![S{+}1]\!],\ \text{(O:C)}[\![S{+}1]\!],$$
$$\neg\exists\ \text{W: (O[M.sc}\twoheadrightarrow\text{W]}[\![S]\!] \wedge \text{W} \neq \text{V)},$$
$$\neg\exists\ \text{D: (O:C)}[\![S{+}1]\!] \wedge \text{(D::C)}[\![S{+}1]\!].$$

$$\text{C'[M.sci}\twoheadrightarrow\text{V]}[\![S{+}1]\!] \leftarrow \text{C[M.sci}\twoheadrightarrow\text{V]}[\![S{+}1]\!],\ \text{(C'::C)}[\![S{+}1]\!],$$
$$\neg\exists\ \text{W: (C'[M.sci}\twoheadrightarrow\text{W]}[\![S]\!] \wedge \text{W} \neq \text{V)},$$
$$\neg\exists\ \text{D: (O:C)}[\![S{+}1]\!] \wedge \text{(D::C)}[\![S{+}1]\!].$$

$$\text{O[M.mvd}\twoheadrightarrow\text{V]}[\![S{+}1]\!] \leftarrow \text{C[M.mvi}\bullet\!\!\twoheadrightarrow\text{V]}[\![S{+}1]\!],\ \text{(O:C)}[\![S{+}1]\!],$$
$$((\neg\exists\ \text{W: O[M.mvd}\twoheadrightarrow\text{W]}[\![S]\!]) \vee$$
$$\forall\ \text{W: O[M.mvd}\twoheadrightarrow\text{W]}[\![S]\!] \leftrightarrow \text{C[M.mvi}\ \bullet\!\!\twoheadrightarrow\text{W]}[\![S{-}1]\!]),$$
$$\neg\exists\ \text{D: (O:D)}[\![S{+}1]\!] \wedge \text{(D::C)}[\![S{+}1]\!].$$

$$\text{C'[M.mvi}\twoheadrightarrow\text{V]}[\![S{+}1]\!] \leftarrow \text{C[M.mvi}\bullet\!\!\twoheadrightarrow\text{V]}[\![S{+}1]\!],\ \text{(C'::C)}[\![S{+}1]\!],$$
$$((\neg\exists\ \text{W: C'[M.mvi}\twoheadrightarrow\text{W]}[\![S]\!]) \vee$$
$$\forall\ \text{W: C'[M.mvi}\twoheadrightarrow\text{W]}[\![S]\!] \leftrightarrow \text{C[M.mvi}\ \bullet\!\!\twoheadrightarrow\text{W]}[\![S{-}1]\!]),$$
$$\neg\exists\ \text{D: (C'::D)}[\![S{+}1]\!] \wedge \text{(D::C)}[\![S{+}1]\!].$$

Example: Win-move game

```
game.S[win ->> X] :- move(X,Y), not game.T[win->>Y], S = T + 1, T.ready[].
game[win->>X] :- game.S[win->>X], S:final.

game[undef->>X] :- game.T[win->>X], not game.S[win->>X], S:final,
                   T:state, S = T + 1.

game[lost->>X] :- X:dom, not game.T[win->>X], S:final, T:state, S = T + 1.

% State sequence (dropped)
% facts

a:dom. b:dom. c:dom. d:dom.
move(a,b).  move(b,a).  move(b,c).  move(c,d).
?- sys.eval[].
?- game[V ->> X].
```

Example: Well-founded Inheritance

Nixon Diamond

```
republican[policy *-> hawk].
quaker[policy *-> pacifist].

nixon:quaker.
nixon:republican.

?- sys.eval[].

?- nixon[policy -> P].
```

Example: Nixon Diamond

```
(republican.S)[(policy.sci) ->> hawk] :- S:state.
(quaker.S)[(policy.sci) ->> pacifist] :- S:state.

(nixon.S):(quaker.S) :- S:state.
(nixon.S):(republican.S) :- S:state.

% block only potential inheritance
(O.S)[blocked@(M.sc) ->> V] :-
   O.S[(M.sc) ->> V], O.S[(M.sc) ->> W], not V = W, S:state.

(O.T)[(M.sc) ->> V] :-
      (C.T)[(M.sci) ->> V], (O.T):(C.T),
      not (O.S)[blocked@(M.sc) ->> V], S:state, T = S + 1.

% state sequence (dropped)

?- S:final.
?- (nixon.S)[(policy.sc) ->> P], S:state.
```

Example: Nixon Diamond

Result:

$$\{P \mid \text{(nixon.0)[(policy.sc)}\twoheadrightarrow\text{P]}\} =$$

$$\{P \mid \text{(nixon.2)[(policy.sc)}\twoheadrightarrow\text{P]}\} = \emptyset$$

$$\{P \mid \text{(nixon.1)[(policy.sc)}\twoheadrightarrow\text{P]}\} = \{\text{hawk,pacifist}\}$$

$$\mathbf{W}(\text{nixon[(policy.sc)}\twoheadrightarrow\text{hawk]}) = \text{undefined}$$

$$\mathbf{W}(\text{nixon[(policy.sc)}\twoheadrightarrow\text{pacifist]}) = \text{undefined}$$

Re-Transformation to original signature:

$$\mathbf{W}(nixon[policy.sc{\rightarrow}hawk]) =$$

$$\mathbf{W}(nixon[policy.sc{\rightarrow}pacifist]) = \text{undefined}$$

Contributions

- Well-Founded Semantics for DOOD Languages
- Negation
- Well-Founded Inheritance