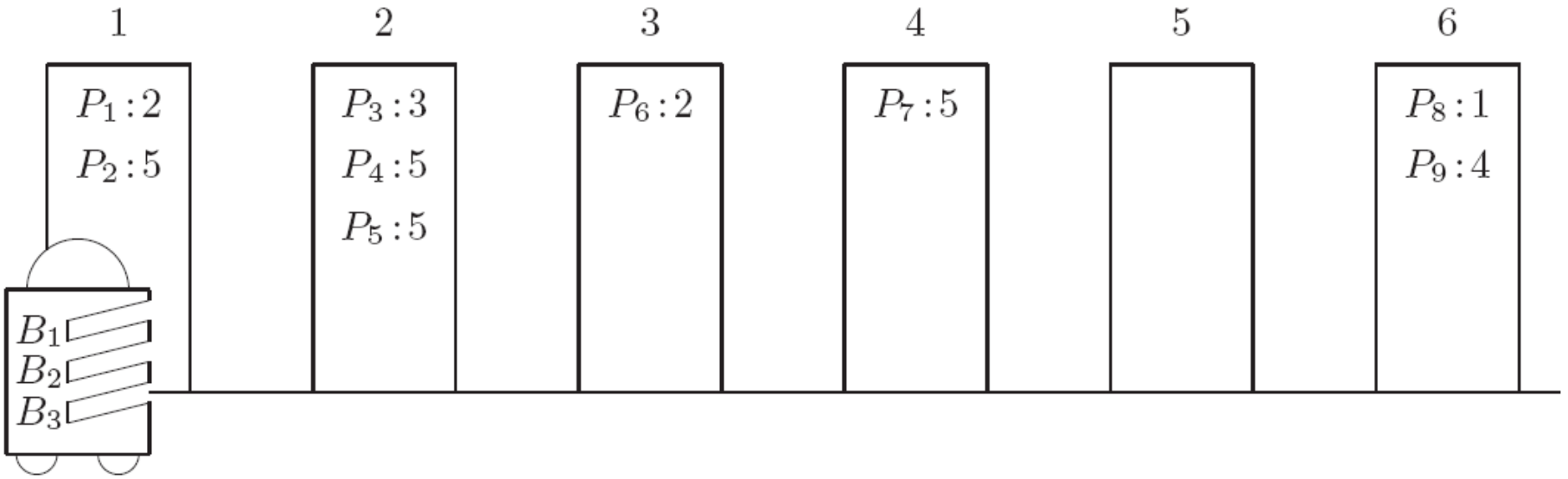
Procedural Action Programs

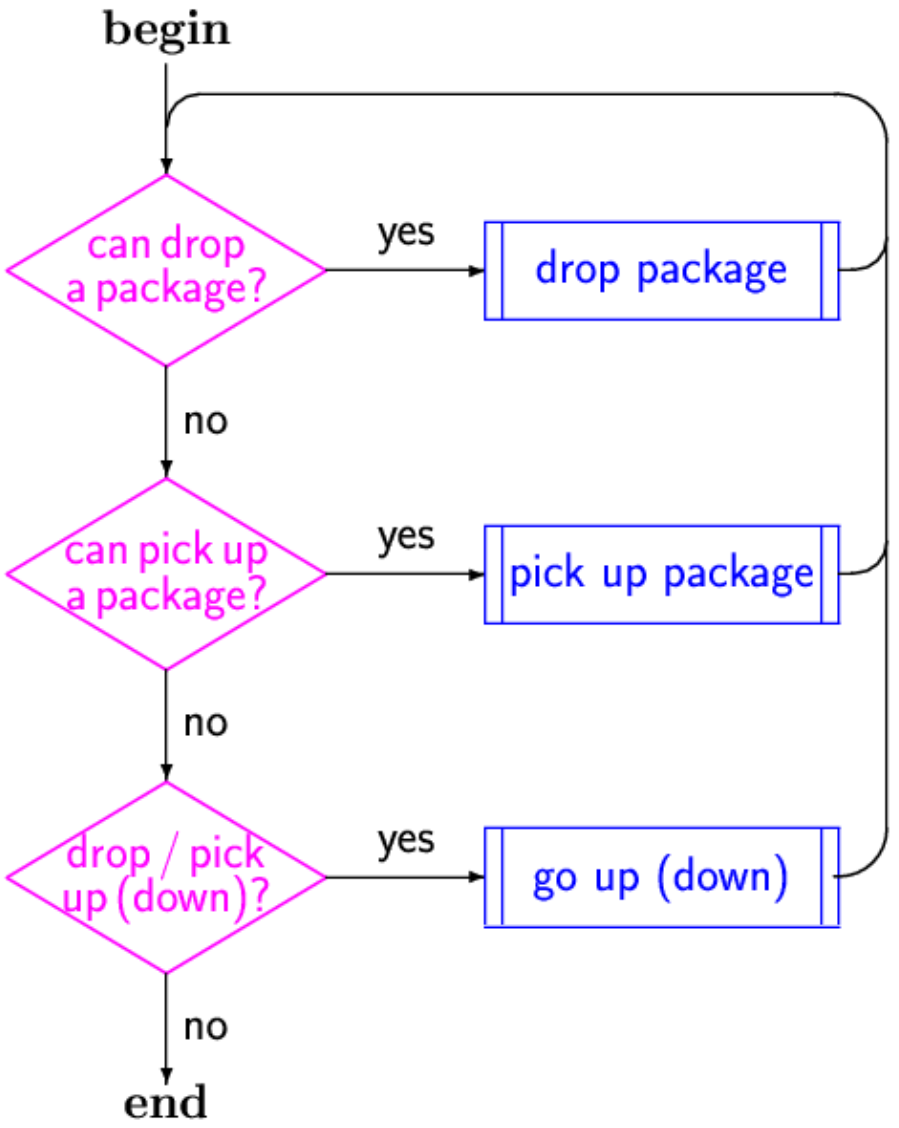
- GOLOG (“Algol in Logic”)
- Semantics: Fluents, Actions, and Effects
- Semantics for GOLOG programs
- A GOLOG Interpreter in Prolog

Example: A Mail Delivery Problem



The task of the robot is to pick up and deliver the 9 packages using its three mail bags.

A Simple Algorithm for the Mailbot



Mail Delivery: Fluents and Actions

Symbol	Type
At	ROOM $\mapsto$ FLUENT
Empty	BAG $\mapsto$ FLUENT
Carries	BAG $\times$ PACKAGE $\times$ ROOM $\mapsto$ FLUENT
Request	PACKAGE $\times$ ROOM $\times$ ROOM $\mapsto$ FLUENT

Symbol	Type
Go	DIRECTION $\mapsto$ ACTION
Pick	PACKAGE $\times$ BAG $\mapsto$ ACTION
Drop	BAG $\mapsto$ ACTION

GOLOG Commands

Command	Meaning
nil	empty program
a	primitive action
ϕ?	test
δ_1 ; δ_2	sequential composition
δ_1 | δ_2	nondeterministic choice of sub-program
πx.δ(x)	nondeterministic choice of argument
δ*	nondeterministic iteration
$p(\vec{t})$	procedure call
if ϕ **then** δ_1 **else** δ_2 **endIf**	conditional
while ϕ **do** δ **endWhile**	loop

GOLOG Programs

proc $p_1(\vec{v}_1)$

δ_1

endProc;

$\vdots$

proc $p_n(\vec{v}_n)$

δ_n

endProc;

δ

A GOLOG Program for the Mailbot (1)

proc *Control*
  while $(\exists b)\ \neg Empty(b) \vee (\exists p, r_1, r_2)\ Request(p, r_1, r_2)$ **do**
    if $(\exists b, p, r)\ (Carries(b, p, r) \wedge At(r))$ **then**
      Deliver
    else
      if $(\exists b, p, r_1, r_2)\ (Empty(b) \wedge Request(p, r_1, r_2) \wedge At(r_1))$ **then**
        Collect
      else
        Continue
      endif
    endif
  endWhile
endProc;

A GOLOG Program for the Mailbot (2)

proc *Deliver*

$\pi b.\ Drop(b)$

endProc;

proc *Collect*

$\pi b.\ \pi p.\ Pick(p, b)$

endProc;

A GOLOG Program for the Mailbot (3)

proc *Continue*

if $(\exists b)\ \neg Empty(b)$ **then**

$\pi b.\ \pi p.\ \pi r.\ \pi r'.\ (At(r)\ ?\ ;\ Carries(b, p, r')\ ?\ ;$

if $r < r'$ **then** $Go(Up)$ **else** $Go(Down)$ **endif**)

else

$\pi p.\ \pi r.\ \pi r_1.\ \pi r_2.\ (At(r)\ ?\ ;\ Request(p, r_1, r_2)\ ?\ ;$

if $r < r_1$ **then** $Go(Up)$ **else** $Go(Down)$ **endif**)

endif

endProc;

The Complete Program

proc *Deliver*

…

proc *Collect*

…

proc *Continue*

…

proc *Control*

…

endProc;

Control

Online- vs. Offline-Execution

Online-execution

- Actions are performed immediately
- Agent commits to every nondeterministic choice
- Program may not be completed successfully even though a successful run exists

Offline-execution

- Program is run in simulation first
- Allows to find a successful run if it exists; compare different runs to find the shortest; etc.
- Practically impossible for large programs with many nondeterministic operations

Semantics:
Conditions and Effects of Actions

A General Action Calculus

Definition 3.3.1

A domain signature is a finite, sorted logic language which includes the sorts FLUENT, ACTION, and TIME along with the predicates

$$<: \text{TIME} \times \text{TIME}$$

$$\mathit{Holds}: \text{FLUENT} \times \text{TIME}$$

$$\mathit{Poss}: \text{ACTION} \times \text{TIME} \times \text{TIME}$$

$s \leq t$ will be used as abbreviation for $s < t \vee s = t$.

State Formulas

Definition 3.3.2

A state formula in variables $\vec{t}$ is a first-order formula $\Phi[\vec{t}]$ in which the elements of $\vec{t}$ occur free and such that

- for each occurence of *Holds*(*f*, *t*) in Φ we have $t \in \vec{t}$;
- predicate *Poss* does not occur in Φ.

Example: State Formula

$$\begin{aligned} Holds(f, s) \equiv \quad & f = At(1) \vee f = Empty(B_1) \vee f = Empty(B_2) \vee f = Empty(B_3) \vee \\ & f = Request(P_1, 1, 2) \vee f = Request(P_2, 1, 5) \vee \ldots \vee \\ & f = Request(P_8, 6, 1) \vee f = Request(P_9, 6, 4) \end{aligned}$$

Example Time Structure: Linear Time

$$(\forall t)\ 0 \leq t$$

$$(\forall s, t)\ s < t \vee t < s \vee s = t$$

$$(\forall r, s, t)\ r < s \wedge s < t \supset r < t$$

$$Poss(Go(x, y, v), s, t) \equiv Holds(At(x), s) \wedge t = s + \frac{1}{v} \bullet Distance(x, y)$$

Example Time Structure: Branching Time

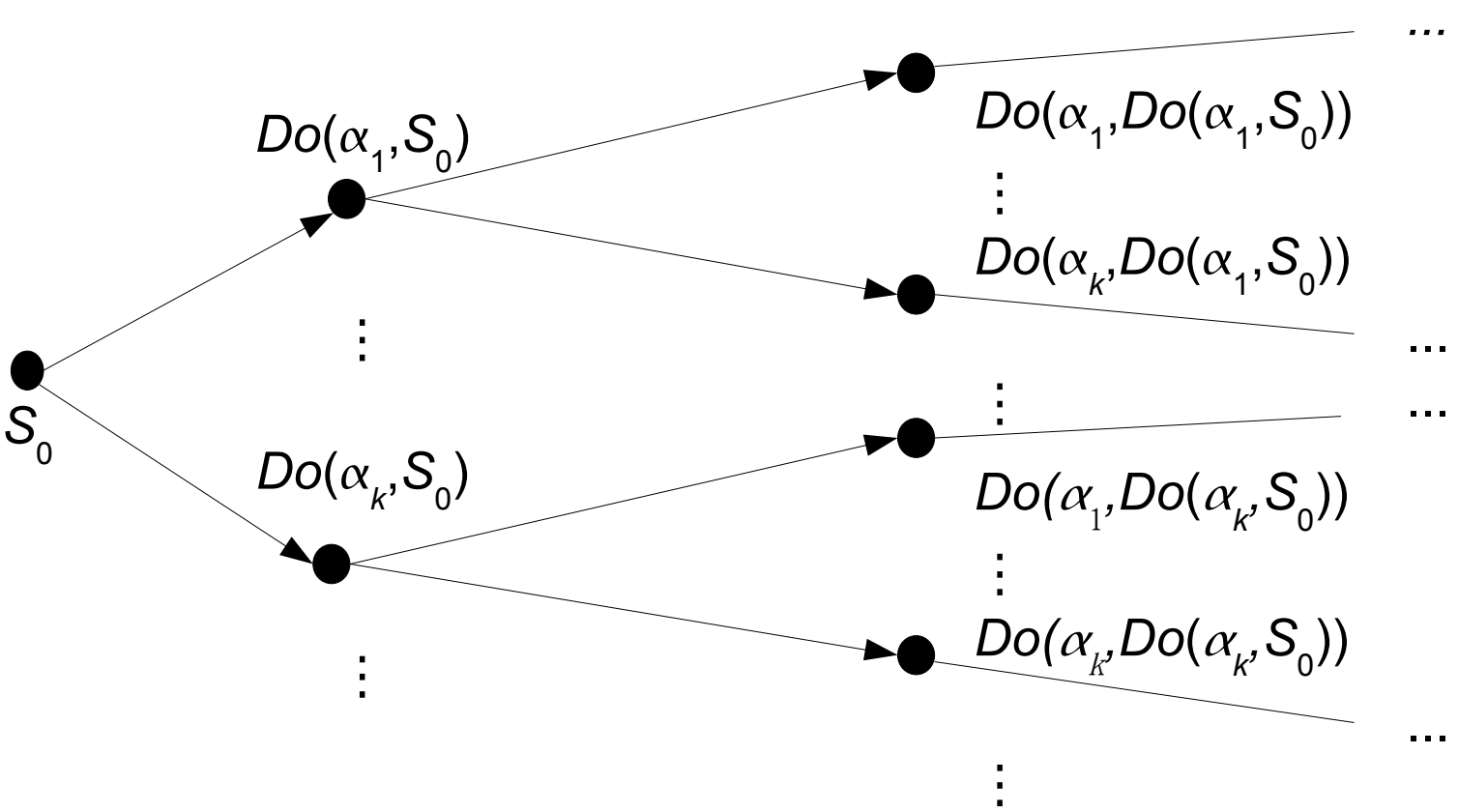
$(\forall s)\ S_0 \leq s$

$(\forall a, a', s, s')(Do(a, s) = Do(a', s') \supset a = a' \wedge s = s')$

$(\forall a, s, s')(s < Do(a, s') \equiv s \leq s')$

Situation Tree

$s=Do(Drop(B_1),Do(Go(Up),Do(Pick(P_2,B_2),Do(Pick(P_1,B_1),S_0))))$



Precondition Axioms

Definition 3.3.3

A precondition axiom is of the form

$$Poss(A(\vec{x}), s, t) \equiv \pi_A[s]$$

where $\pi_A[s]$ is a state formula in s with free variables among s, t, $\vec{x}$.

Example: Precondition Axioms

$$Poss(Go(d), s, t) \equiv t = Do(Go(d), s) \wedge$$
$$(\exists r)(Holds(At(r), s) \wedge [d = Up \wedge r < 6 \vee d = Down \wedge r > 1])$$

$$Poss(Pick(p, b), s, t) \equiv t = Do(Pick(p, b), s) \wedge$$
$$(\exists r, r')(Holds(At(r), s) \wedge Holds(Request(p, r, r'), s) \wedge$$
$$Holds(Empty(b), s))$$

$$Poss(Drop(b), s, t) \equiv t = Do(Drop(b), s) \wedge$$
$$(\exists p, r)(Holds(At(r), s) \wedge Holds(Carries(b, p, r), s))$$

Frame Problem (McCarthy and Hayes, 1969)

The frame problem arises because mere effect formulas do not suffice to draw conclusions about unchanged fluents.

$$Poss(Pick(p,b),s) \supset (\exists r,r')\,(Holds(Request(p,r,r'),s) \wedge \neg Holds(Empty(b),Do(Pick(p,b),s))$$
$$\wedge \neg Holds(Request(p,r,r'),Do(Pick(p,b),s))$$
$$\wedge Holds(Carries(p,r,r'),Do(Pick(p,b),s)))$$

Suppose given $Holds(At(1),S_0) \wedge Holds(Empty(B_1),S_0) \wedge Holds(Request(P_1,1,2),S_0)$

The mere effect axiom (with $\{p/P_1,b/B_1,s/S_0\}$) does not entail
$Holds(At(1),Do(Pick(P_1,B_1),S_0))$

Effect Axioms

Definition 3.3.3 (cont'd)

An effect axiom is of the form

$$Poss(A(\vec{x}), s, t) \supset \Upsilon_1[s, t] \vee \ldots \vee \Upsilon_k[s, t]$$

where $k \geq 1$ and each $\Upsilon_i[s, t]$ $(1 \leq i \leq k)$ is a formula of the form

$$(\exists \vec{y}_i)(\Phi_i[s] \wedge (\forall f)[\Gamma_i^+[s, t] \supset Holds(f, t)]$$

$$\wedge (\forall f)[\Gamma_i^-[s, t] \supset \neg Holds(f, t)])$$

in which $\Phi_i[s]$ is a state formula in s with free variables among s, $\vec{x}$, $\vec{y}_i$, and both $\Gamma_i^+[s, t]$ and $\Gamma_i^-[s, t]$ are state formulas in s, t with free variables among f, s, t, $\vec{x}$, $\vec{y}_i$.

Example: Effect Axiom (1)

$$
\begin{aligned}
&Poss(Go(d), s, t) \supset \\
&\quad d = Up \wedge (\exists r)(Holds(At(r), s) \wedge \\
&\qquad (\forall f)[f = At(r + 1) \vee (Holds(f, s) \wedge f \neq At(r)) \supset Holds(f, t)] \\
&\qquad \wedge \\
&\qquad (\forall f)[f = At(r) \vee (\neg Holds(f, s) \wedge f \neq At(r + 1)) \supset \neg Holds(f, t)]) \\
&\quad \vee \\
&\quad d = Down \wedge (\exists r)(Holds(At(r), s) \wedge \\
&\qquad (\forall f)[f = At(r - 1) \vee (Holds(f, s) \wedge f \neq At(r)) \supset Holds(f, t)] \\
&\qquad \wedge \\
&\qquad (\forall f)[f = At(r) \vee (\neg Holds(f, s) \wedge f \neq At(r - 1)) \supset \neg Holds(f, t)])
\end{aligned}
$$

Example: Effect Axiom (2)

$$
\begin{aligned}
&\mathit{Poss}(\mathit{Pick}(p, b), s, t) \supset \\
&\quad (\exists r, r')(\mathit{Holds}(\mathit{Request}(p, r, r'), s) \wedge \\
&\qquad (\forall f)[f = \mathit{Carries}(b, p, r') \vee (\mathit{Holds}(f, s) \wedge f \neq \mathit{Empty}(b) \wedge f \neq \mathit{Request}(p, r, r')) \\
&\qquad\qquad \supset \mathit{Holds}(f, t)] \\
&\qquad \wedge \\
&\qquad (\forall f)[f = \mathit{Empty}(b) \vee f = \mathit{Request}(p, r, r') \vee (\neg\mathit{Holds}(f, s) \wedge f \neq \mathit{Carries}(b, p, r')) \\
&\qquad\qquad \supset \neg\mathit{Holds}(f, t)])
\end{aligned}
$$

Example: Effect Axiom (3)

$$
\begin{aligned}
&Poss(Drop(b), s, t) \supset \\
&\quad (\exists p, r)(Holds(Carries(b, p, r), s) \wedge \\
&\qquad (\forall f)[f = Empty(b) \vee (Holds(f, s) \wedge f \neq Carries(b, p, r)) \supset Holds(f, t)] \\
&\qquad \wedge \\
&\qquad (\forall f)[f = Carries(b, p, r) \vee (\neg Holds(f, s) \wedge f \neq Empty(b)) \supset \neg Holds(f, t)])
\end{aligned}
$$

Signatures with Functional Fluents

It is often more natural and economic to describe the state space by functional fluents instead of relational ones.

Definition 3.3.4

A functional domain signature is a finite, sorted logic language which includes the sorts FLUENT, ACTION, TIME, and VALUE along with the predicates

$$<: \text{TIME} \times \text{TIME}$$

$$\mathit{Poss}: \text{ACTION} \times \text{TIME} \times \text{TIME}$$

and the function

$$\mathit{Val}: \text{FLUENT} \times \text{TIME} \dot{} \text{VALUE}$$

Functional Domain Axiomatizations (1)

Definition 3.3.4 (cont'd)

A state formula in variables $\vec{t}$ is a first order formula $\Phi[\vec{t}]$ in which the elements of $\vec{t}$ occur free and such that

- for each occurence of $Val(f, t)$ in Φ we have $t \in \vec{t}$;
- predicate *Poss* does not occur in Φ.

A precondition axiom is of the form

$$Poss(A(\vec{x}), s, t) \equiv \pi_A[s]$$

where $\pi_A[s]$ is a state formula in s with free variables among s, t, $\vec{x}$.

Functional Domain Axiomatizations (2)

Definition 3.3.4 (cont'd)

An effect axiom is of the form

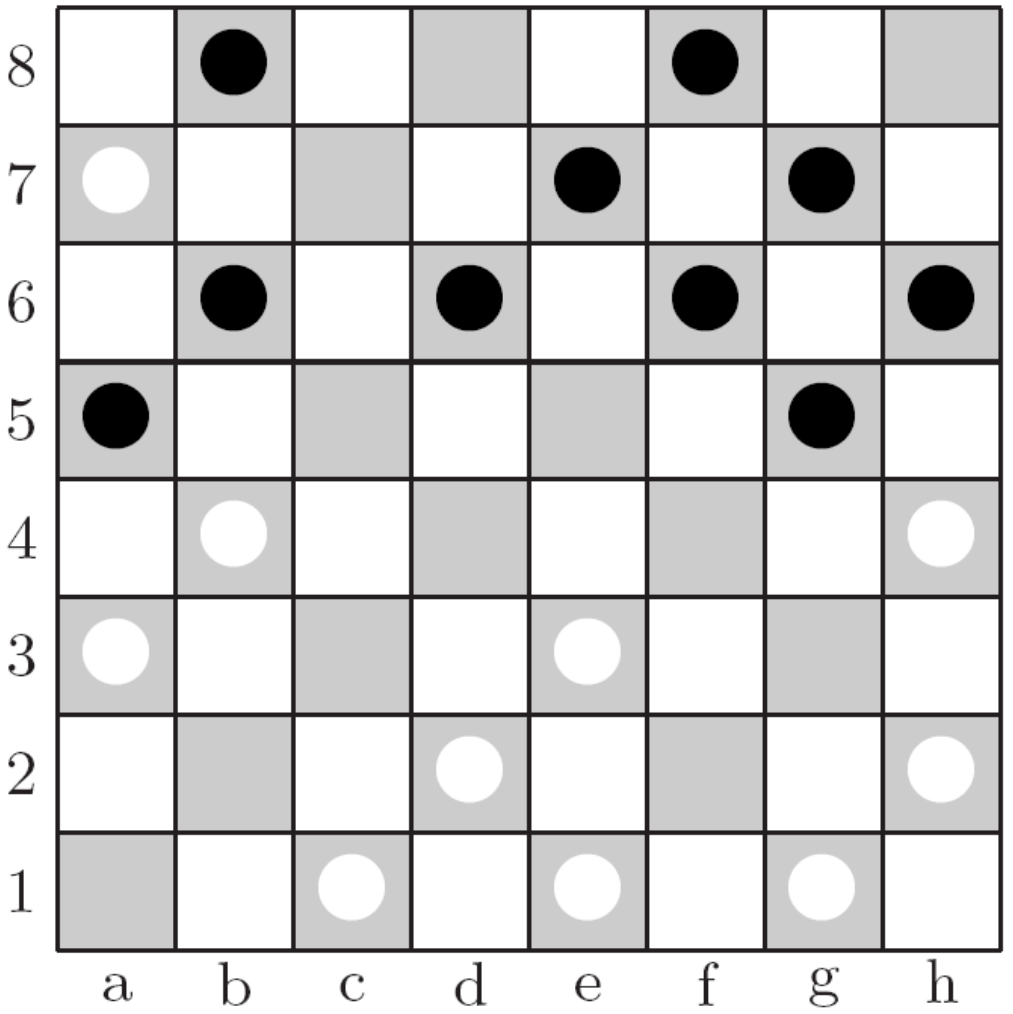
$$Poss(A(\vec{x}), s, t) \supset \Upsilon_1[s, t] \vee \ldots \vee \Upsilon_k[s, t]$$

where $k \geq 1$ and each $\Upsilon_i[s, t]$ $(1 \leq i \leq k)$ is a formula of the form

$$(\exists \vec{y}_i)(\Phi_i[s] \wedge (\forall f, v)[\Gamma_i[s, t] \supset Val(f, t) = v])$$

in which $\Phi_i[s]$ is a state formula in s with free variables among, s, $\vec{x}$, $\vec{y}_i$, and $\Gamma_i[s, t]$ is a state formula in s, t with free variables among f, s, t, $\vec{x}$, $\vec{y}_i$, v.

Example: Checkers



Example: Fluents and Actions for Checkers

Symbol	Type	Range
Cell	FILE × ROW ↦ FLUENT	{*Blank*, *White*, *WhiteKing*, *Black*, *BlackKing*}
Control	↦ FLUENT	{*White*, *Black*}

Symbol	Type
Move	FILE × ROW × FILE × ROW ↦ ACTION

Val(*Control*, 0) = *Black* ∧

Val(*Cell*(*a*,1), 0) = *White* ∧ ... ∧ *Val*(*Cell*(*h*,8),0) = *Black*

Example: Precondition Axiom for Checkers

$$Poss(Move(x_1, y_1, x_2, y_2), s, t) \equiv$$

$$Occurs(Move(x_1, y_1, x_2, y_2), s) \wedge t = s + 1 \wedge$$

$$[LegalWhiteMove \vee LegalBlackMove \vee LegalKingMove \vee LegalJump]$$

where

$$LegalWhiteMove \stackrel{def}{=} Val(Control, s) = White \wedge Val(Cell(x_1, y_1), s) = White \wedge$$

$$Val(Cell(x_2, y_2), s) = Blank \wedge y_1 < 8 \wedge y_2 = y_1 + 1 \wedge$$

$$NeighborFiles(x_1, x_2)$$

etc.

Example: Effect Axioms for Checkers

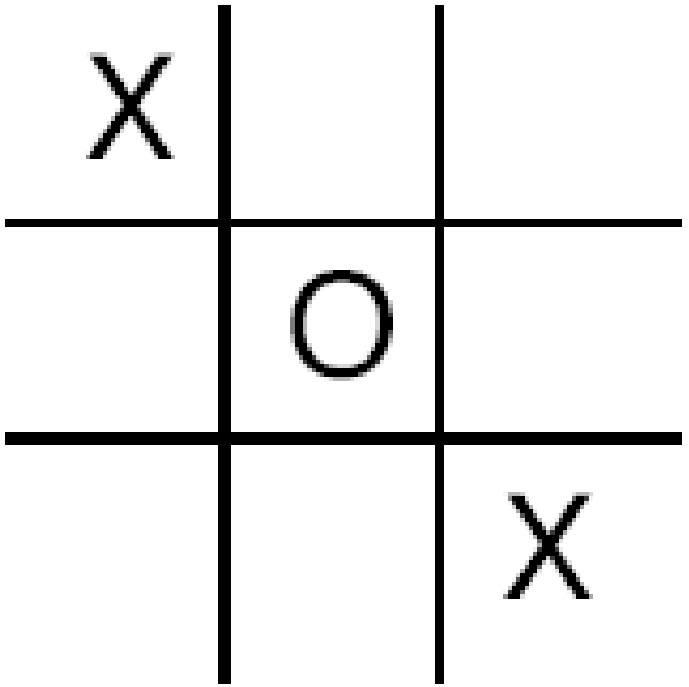
$$Poss(Move(x_1, y_1, x_2, y_2), s, t) \supset LegalWhiteMove \wedge WhiteMoveUpdate \vee$$
$$LegalBlackMove \wedge BlackMoveUpdate \vee$$
$$LegalKingMove \wedge KingMoveUpdate \vee$$
$$LegalJump \wedge JumpUpdate$$

where

$$WhiteMoveUpdate \stackrel{def}{=} (\forall f, v)[f = Cell(x_1, y_1) \wedge v = Blank \vee$$
$$f = Cell(x_2, y_2) \wedge (y_2 < 8 \wedge v = White \vee y_2 = 8 \wedge v = WhiteKing) \vee$$
$$f = Control \wedge v = Black \vee$$
$$f \neq Cell(x_1, y_1) \wedge f \neq Cell(x_2, y_2) \wedge f \neq Control \wedge v = Val(f, s) \supset Val(f, t) = v)$$

etc.

Other Description Techniques: The Game Description Language (GDL)



Fluents

```
cell(number,number,symbol)
control(player)
```

Actions

```
mark(number,number)
noop
```

Precondition Axioms in GDL

```
legal(P,mark(X,Y)) <=
    true(cell(X,Y,blank)) ∧
    true(control(P))

legal(xplayer,noop) <=
    true(cell(X,Y,blank)) ∧
    true(control(oplayer))

legal(oplayer,noop) <=
    true(cell(X,Y,blank)) ∧
    true(control(xplayer))
```

Effect Axioms in GDL

```
next(cell(M,N,xsymb)) <= does(xplayer,mark(M,N))
next(cell(M,N,osymb)) <= does(oplayer,mark(M,N))
next(control(xplayer)) <= true(control(oplayer))
next(control(oplayer)) <= true(control(xplayer))
```

The frame problem in GDL: Suppose given

```
true(cell(1,1,xsymb)) ∧ true(cell(1,2,blank)) ∧ ...
does(oplayer,mark(3,2)) ∧ does(xplayer,noop)
```

The mere effect axioms do not entail

```
next(cell(1,1,xsymb)), next(cell(1,2,blank)), ...
```

Coping with the Frame Problem in GDL

```
next(cell(M,N,W)) <=
    true(cell(M,N,W)) ∧
    distinct(W,blank)

next(cell(M,N,blank)) <=
    true(cell(M,N,blank)) ∧
    does(P,mark(J,K)) ∧
    (distinct(M,J) ∨ distinct(N,K))
```

These are also known as frame axioms.

Characteristics of GDL

- Binary time structure (`true, next`)
- Actions are collections of individual moves
- Explicit frame axioms are used