Relational and First-Order Decision-Theoretic Planning:

Foundations and Future Directions

Scott Sanner

`ssanner@cs.toronto.edu`

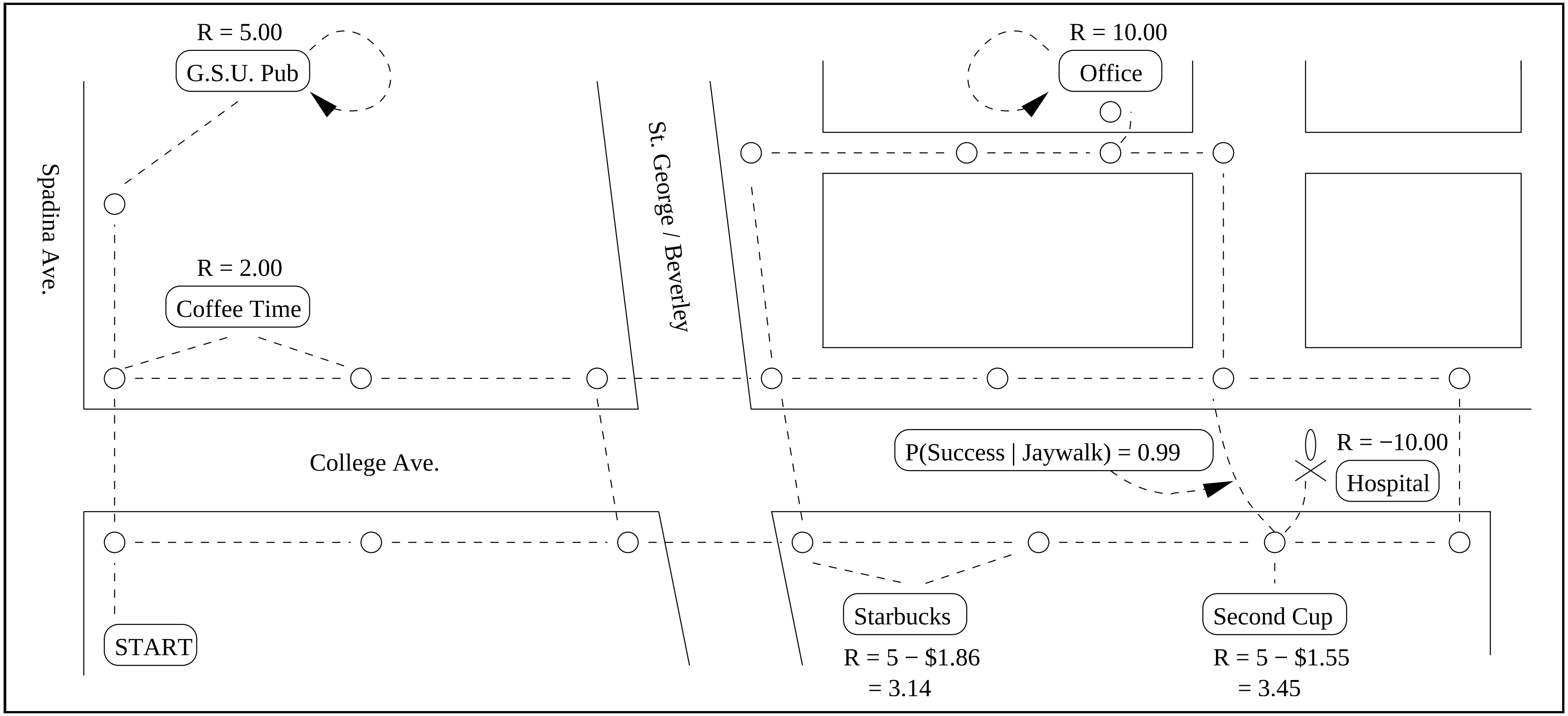
Depth-Oral Presentation

Department of Computer Science

University of Toronto

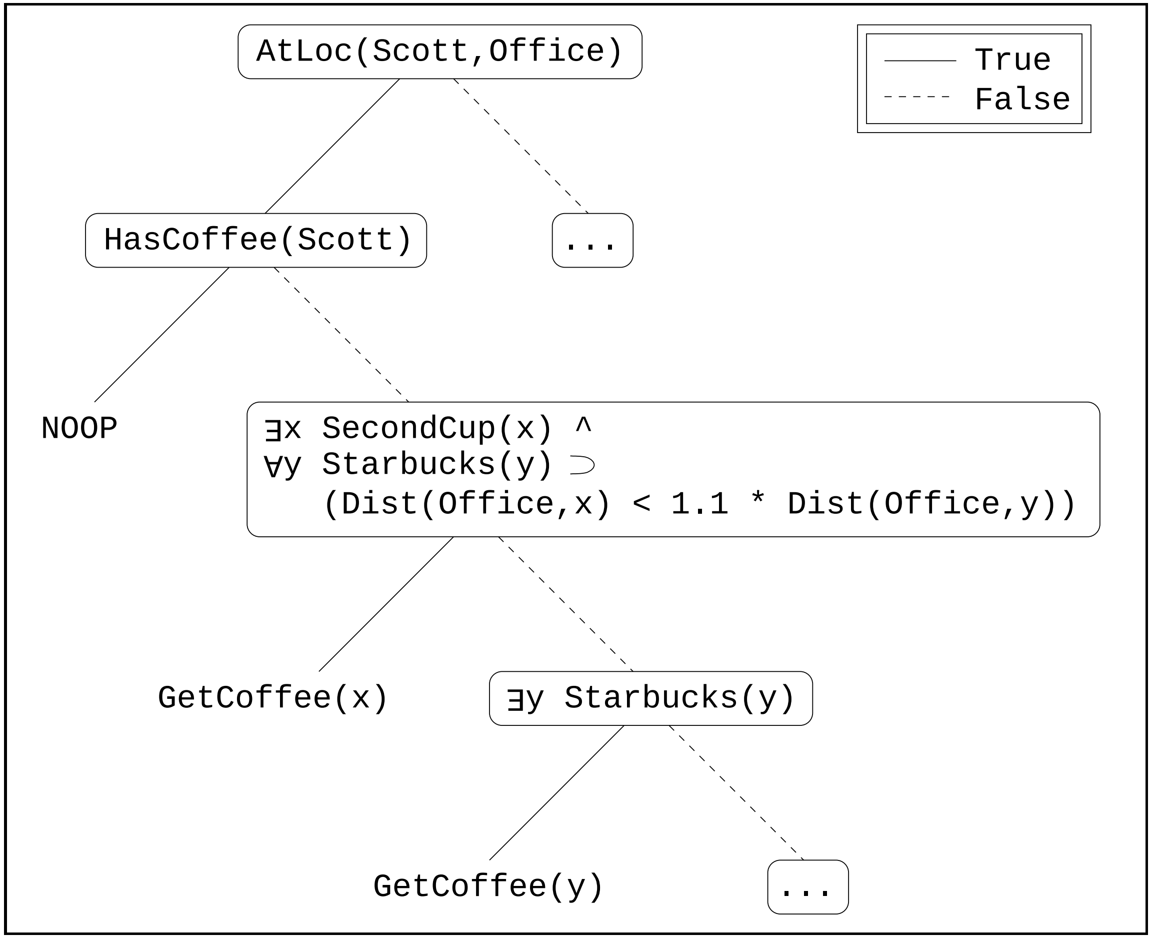
Importance of DT Planning ☺

Example: Planning the walk to the office.

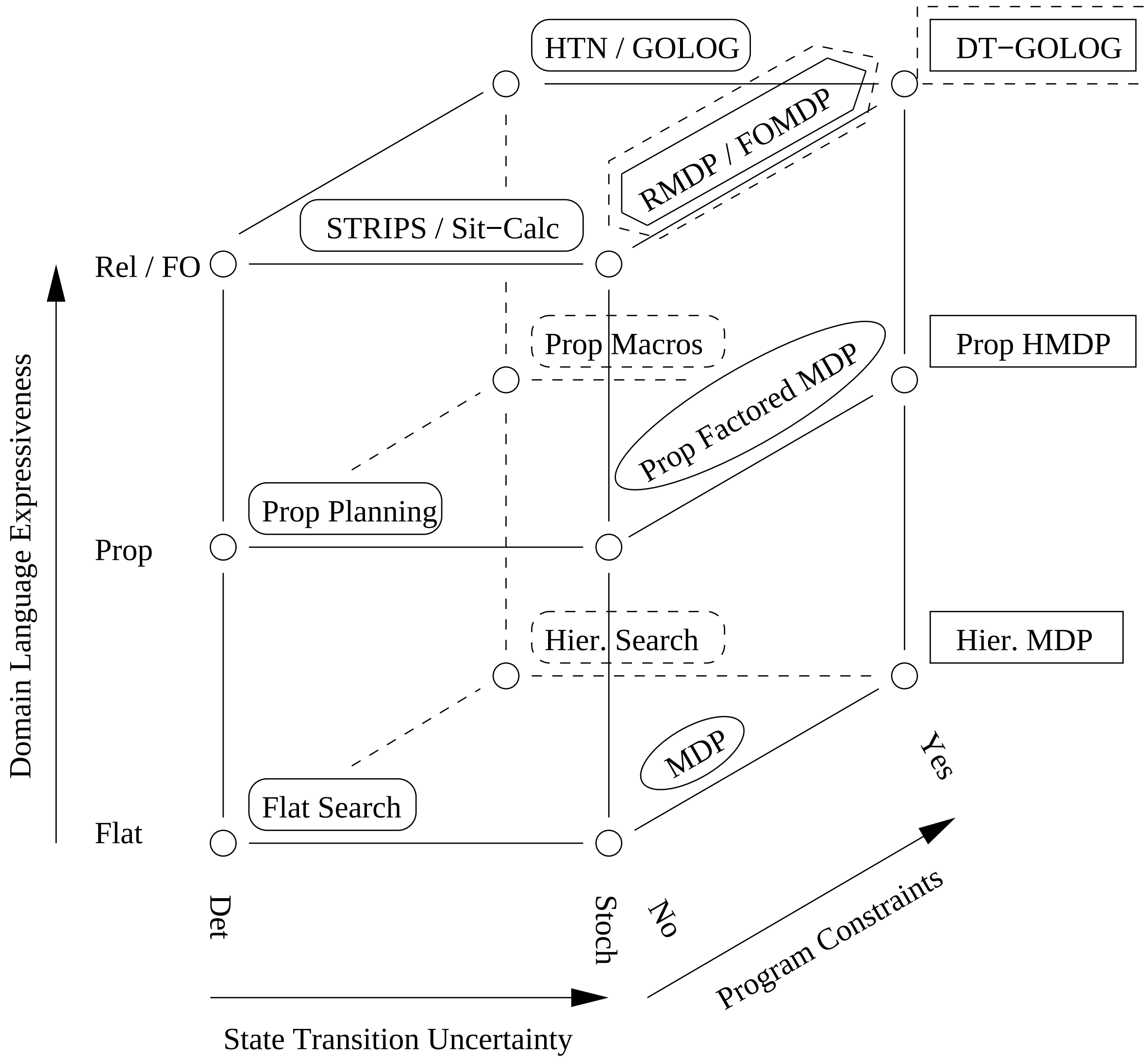


Importance of FO/Rel. Representation

But how would we plan for all possible worlds?



Presentation Overview



Foundations:

Deterministic planning

MDPs and structured representations

Relational and first−order MDPs

Hierarchy and program constraints

Future directions:

Structured representations in FOMDPs

Approximation in FOMDPs

FOMDPs and program constraints

Det. Planning: Relational and FO

- STRIPS and PDDL
  - States: $\{On(b_1, b_2), On(b_2, b_3)\}$
  - Actions:

```
(:action stack
 :parameters (?a ?b)
 :precondition (and (forall (?c) (not (on ?c ?a)) ... ))
 :effect       (and (forall (?c)
                       (when (on ?a ?c)
                             (not (on ?a ?c))))
                       (on ?a ?b)))
```

  - Problem: Dom.: b_1, b_2; Init. state; Goal: $\{On(x, y) \wedge On(y, z)\}$
- Situation calculus
  - Actions: $stack(b_2, b_1)$, Situations: s_0, $do(stack(b_2, b_1), s_0)$, Fluents: $On(b_2, b_1, s_0)$, SSAs: $F(\vec{x}, do(a, s)) \equiv \Phi_F(\vec{x}, a, s)$
  - Problem: UNA, SSAs, action preconditions, initial situation

Det. Planning: Program constraints

GOLOG (LRLLS, 1997)

- Program δ restricts execution: $[a; b; [c|d]]^*$
- Specify execution using $Do(\delta, s, s')$
- $Do(\delta, s, s')$ defined inductively on first argument:
  - *Primitive actions (base case)*
  - *Test actions*
  - *Sequence of programs*
  - *Nondeterministic* $\{$*program choice*, *choice of arguments*, *iteration*$\}$
  - *Other constructs:* $\{$*if / then*, *while*, *procedures*, *programs*$\}$
- Goal-Oriented Planning: $Axioms \models (\exists s).Do(\delta, S_0, s) \wedge G(s)$

MDP Representation (Informal)

- So far, we have discussed
  - Goal-oriented,
  - Relational / first-order representations, and
  - Program constraints.
- But how do we plan with
  - General reward functions / action costs?
  - Uncertain state transitions?
- One possible solution
  - Markov decision process (MDP) framework,
  - Assume infinite horizon,
  - Maximize expected sum of discounted future rewards.

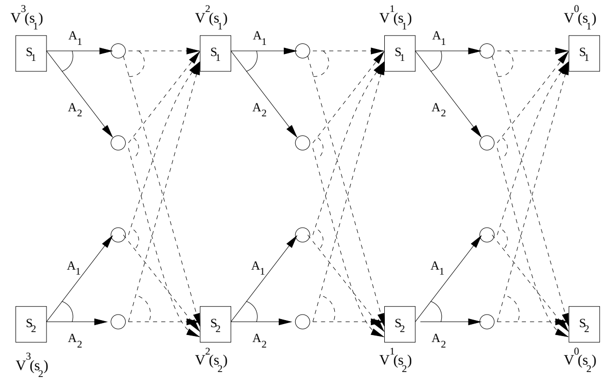
MDP Representation (Formal)

- MDP formally defined as $\langle S, A, T, R \rangle$
  - S: finite set of states
  - A: finite set of actions
  - T: transition function ($T : S \times A \times S \rightarrow [0, 1]$)
  - R: reward function ($R : S \times A \rightarrow \mathbb{R}$)
- Additionally define
  - π: policy mapping from states to actions ($\pi : S \rightarrow A$)
  - r^t: reward at time step t
  - γ: discount factor where $0 \leq \gamma < 1$
  - $V_\pi(s) = E_\pi[\sum_{t=0}^{\infty} \gamma^t \cdot r^t | s_0 = s]$: value of π starting from s
- Goal: find optimal policy π^*
  - $V^*(s) = V_{\pi^*}(s)$ maximizes value over all states

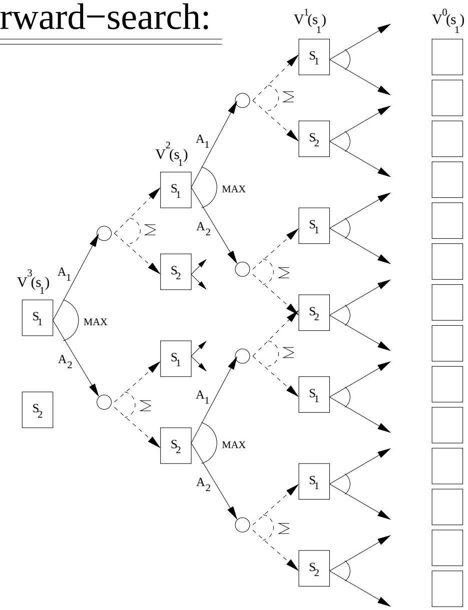
MDP Solution Algorithms

- Dynamic programming
  - Value iteration
  - Policy iteration
  - Modified policy iteration
- Forward-search
- Real-time dynamic programming
- Linear program

Value iteration:



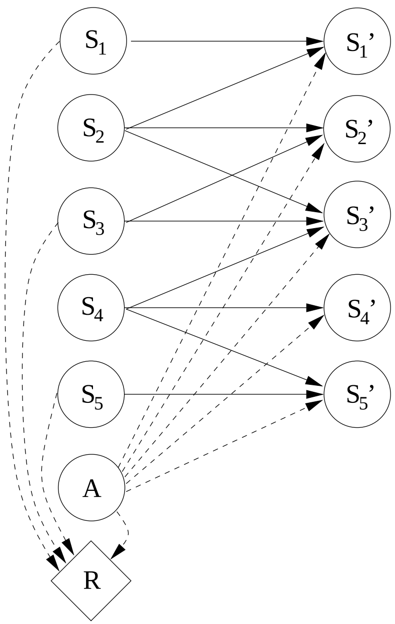
Forward−search:



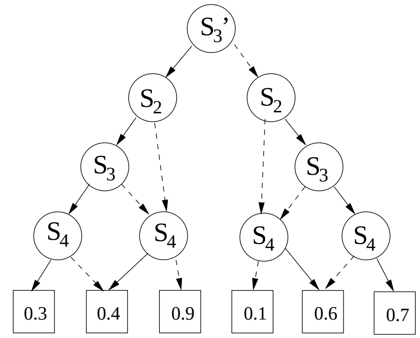
Prop. MDPs and Structured Repr. I

- Factored representation
  - Transition DBN
  - Influence diagram for reward
- Context-specific independence
  - Transition CPT
  - Reward structure
- Algorithms that exploit structure
  - SPI (BDG,1995)
  - SPUDD (HSHB,1999)
  - APRICODD (SHB,2000)

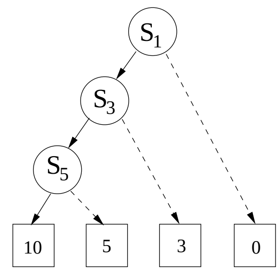
Transition DBN:



ADD for $P(S_3' | S_2, S_3, S_4)$:



ADD for reward:



Prop. MDPs and Structured Repr. II

- Exploit additive structure in value function
- Use linear combination of basis vectors:

$$V = w_1 a_1 + w_2 a_2 + w_3 a_3 \qquad A = \{a_1, a_2, a_3\}$$

- Solve with LP or approximate PI (API):

$$\begin{aligned} \vec{w}^{(t)} &= \arg\min_{\vec{w}} \left\| (A - \gamma T_{\pi^{(t)}} A)\vec{w} - R_{\pi^{(t)}} \right\| \\ \pi^{(t+1)} &= \arg\max_{\pi \in \Pi} \left(R_\pi + \gamma T_\pi A \vec{w}^{(t)} \right) \end{aligned}$$

- (GKP,2001) give compact LP for $\mathcal{L}_\infty$ proj.
- API $\mathcal{L}_2$: (KP,1999), API $\mathcal{L}_\infty$: (GKP,2001), LP $\mathcal{L}_\infty$: (SP,2001)

Relational MDP Representation

- PSTRIPS, PPDDL

```
(:action stack
 :parameters (?a ?b)
 :precondition (and (forall (?c) (and (not (on ?c ?a) ... ))))
 :effect (probabilistic 0.7 (and (forall (?c)
                                    (when (on ?a ?c)
                                          (not (on ?a ?c))))
                                    (on ?a ?b))))
(define
  (:objects block0 block1 block2 ... )
  (:goal (exists (?b1 ?b2) (and (red ?b1) (on ?b1 ?b2))))
```

- Concise representation of DT planning problem
- But action and state space can be extremely large even for small domain instantiations

Relational MDP Algorithms

First-order policy induction (YFG,2002; FYG,2003)

- Algorithm: Perform approximate policy iteration (using decision-list policies)
  - Draw training samples $D^{(t)}$ under a current policy $\pi^{(t)}$
  - Induce a new policy $\pi^{(t+1)}$ from $D^{(t)}$
- Example decision-list policy:

$$\begin{aligned} goal\text{-}on(a,b) \wedge \neg on(a,b) \wedge holding(a) &\longrightarrow putdown(a,b) \\ goal\text{-}on(a,b) \wedge \neg on(a,b) &\longrightarrow pickup(a) \end{aligned}$$

Basis value function approximation (GKGK,2003)

- Algorithm: Compute basis value functions (using LP over sampled, weighted domains)
- Example value function: $V(s) = V_{fo}{}^{otman}(f_1, e_1) + V_{fo}{}^{otman}(f_2, e_2)$

First-order MDPs

- ## Representation
  - Nature’s choice for stochastic actions

    $P(PutdownS(b_1, b_2) \mid Putdown(b_1, b_2), s) = [Wet(b_1, s) : 0.7 \; ; \; \neg Wet(b_1, s) : 0.9$

    $P(PutdownF(b_1, b_2) \mid Putdown(b_1, b_2), s) = [Wet(b_1, s) : 0.3 \; ; \; \neg Wet(b_1, s) : 0.1$

  - Reward and value function case partitions

$$
case \quad [\quad (\exists b_1, b_2).On(b_1, b_2, s) \wedge Red(b_1) : 10 \; ;
$$
$$
\neg((\exists b_1, b_2).On(b_1, b_2, s) \wedge Red(b_1)) \wedge (\exists b_1, b_2).On(b_1, b_2, s) : 5 \; ;
$$
$$
\neg(\exists b_1, b_2).On(b_1, b_2, s) : 0 \quad]
$$

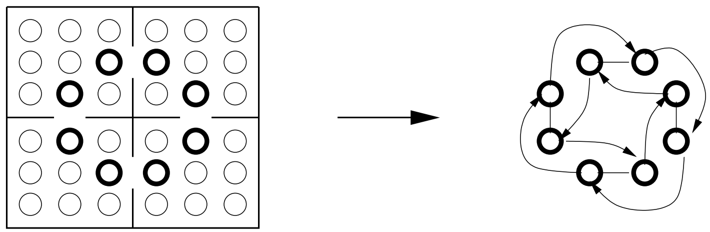
- ## Solution approach
  - Symbolic version of value iteration (BRP,2001)
  - Final ϵ-optimal value function is a case partition

MDPs with Program Constraints

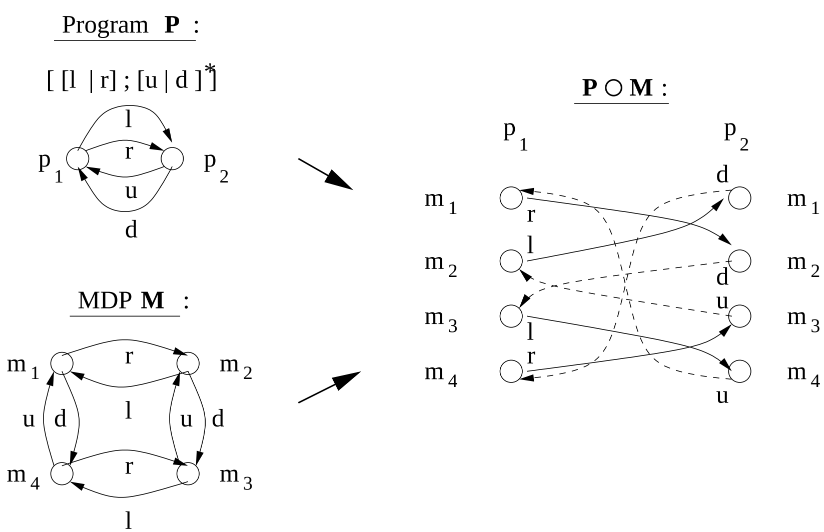
Approaches:

- Markov task decomposition
  (MHKPKDB,1998)
- Macro-actions
  (SPS,1999)
- MDP decomposition / abstraction
  (HMKDB,1998)
- Program constraints
  (PR,1998; AR,2001/2)
- Macro-actions *and* program constraints
  (Diet,1998)

MDP Decomposition / Abstraction:



Program Constraints:

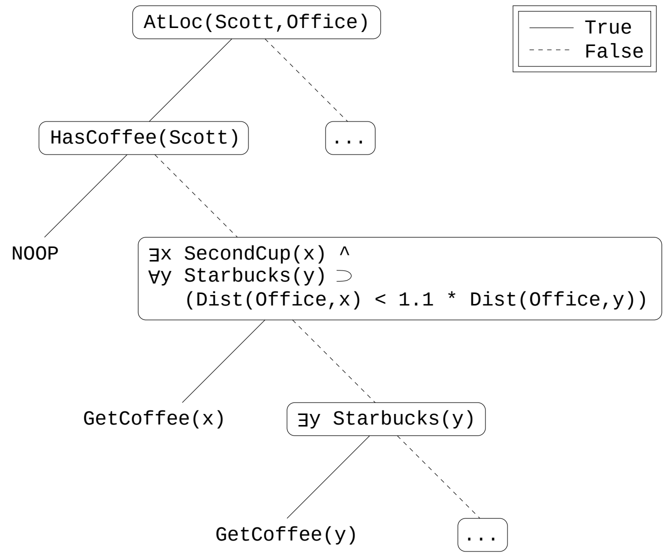


FOMDPs with Program Constraints

- DT-GOLOG representation (BRST,2000)
  - Start with GOLOG's $Do(\delta, s, s')$
  - Generalize to a function representing the decision-theoretically optimal execution of δ: $BestDo(\delta, s) \rightarrow \mathbb{R}$
  - *DT nondeterministic program choice*: $BestDo([\delta_1|\delta_2]) = \max\ \{BestDo(\delta_1)\ ;\ BestDo(\delta_2)\}$
  - *DT nondeterministic choice of arguments*: $BestDo((\pi x)\delta(x)) = \max_x\ \{BestDo(\delta(x))\}$
- Solution approaches
  - Forward-search (BRST,2000)
  - Forward-search with macros (FFL,2003)
  - Real-time dynamic programming (Sout,2001)

Future Directions I

- FOMDPs
  - First-order ADDs and approximation
    - Partitions often have redundant structure
    - Exploit structure for computational and representational efficiency
    - Use APRICODD-style pruning for approximation
  - First-order basis functions and approximation
    - Value function can be approximately additive
    - Exploit structure with weighted FO-basis functions



$$V(s) =$$

$$\begin{array}{llll} w_1 \cdot case & [& (\exists c).\neg R(c,s) \wedge S(c) : 1 \; ; \\ & & (\forall c).S(c) \supset R(c,s) : 0 \;] \\ & + & \\ w_2 \cdot case & [& (\exists c).\neg R(c,s) \wedge T(c) : 1 \; ; \\ & & (\forall c).T(c) \supset R(c,s) : 0 \;] \end{array}$$

Future Directions II

- FOMDPs
  - DBN factored action decomposition
  - Domain constraints
  - First-order counting quantifiers / aggregators

$$Reward = case[(\exists_n b).Red(b) \wedge In(b,P,s) : 10 \; ; \; \neg(\exists_n b).Red(b) \wedge In(b,P,s) : 0 \;]$$

$$P(Running(c,s) \mid NoReboot(c,s) = \frac{\sharp_d . Connected(c,d) \wedge Working(d)}{\sharp_d . Connected(c,d)}$$

- FOMDPs with program constraints
  - First-order DT regression under program constraints
  - Allows FODTR under prog. constraints without initial state knowledge