CS-E4800 Artificial Intelligence

Jussi Rintanen

Department of Computer Science
Aalto University

February 9, 2017

Aalto University

Logic-Based Breadth-First Search

Before SAT-based reachability, previous breakthrough in 1989.

- sets = formulas, relations = formulas
- plugged in in a conventional breadth-first search algorithm
- model-checking and verification (Coudert et al. 1989, Burch et al., 1994)
- formulas as Binary Decision Diagrams

Image Operations

When viewing actions as relations R, the successor states of a set S are obtained with the image operation.

$$img_R(S) = \{s' | s \in S, sRs'\}$$

We will later define a logical image operation when both S and R are represented as formulas.

Analogous pre-image operation defined similarly.

$$preimg_R(S) = \{s | s' \in S, sRs'\}$$

Reachable States by Breadth-First Search

INPUT: I = set of initial states
R = binary relation on states

1. $i := 0$
2. $S_0 := I$
3. $i := i + 1$
4. $S_i := S_{i-1} \cup img_R(S_{i-1})$
5. if $S_i \neq S_{i-1}$ then go to step 3

$S_0, S_1, S_2, \ldots$ consist of states reachable by $\leq 0, \leq 1, \leq 2, \ldots$ actions, respectively
In the end, S_i consists of all reachable states.

Formulas as Data Structure: Relations

Successor states of $\{000, 010, 111\}$ w.r.t. relation $\{(000, 011), (001, 010), (010, 001), (011, 000)\}$?

First Step: Select matching lines from state set and relation by natural join:

$$\begin{array}{c} 0 \\ \hline 000 \\ 010 \\ 111 \end{array} \bowtie \begin{array}{cc} 0 & 1 \\ \hline 000 & 011 \\ 001 & 010 \\ 010 & 001 \\ 011 & 000 \end{array} = \begin{array}{cc} 0 & 1 \\ \hline 000 & 011 \\ 010 & 001 \end{array}$$

Formulas as Data Structure: Relations

Second Step: Project the successor states from the selected subset of the relation

$$\Pi_1 \left(\begin{array}{cc} 0 & 1 \\ \hline 000 & 011 \\ 010 & 001 \end{array} \right) = \begin{array}{c} 1 \\ \hline 011 \\ 001 \end{array}$$

Successor states of $\{000, 010, 111\}$ w.r.t. relation $\{(000, 011), (001, 010), (010, 001), (011, 000)\}$?

They are $\{001, 011\}$.

Relation Operations in Logic

When sets and relations are represented as formulas, how to perform the corresponding relation operations?

relation operation	logical operation
(natural) join	conjunction
projection	$\exists$-abstraction

Natural Join as Conjunction

$$
\begin{array}{c}
0 \\
\hline
000 \\
010 \\
111
\end{array}
\bowtie
\begin{array}{cc}
0 & 1 \\
\hline
000 & 011 \\
001 & 010 \\
010 & 001 \\
011 & 000
\end{array}
=
\begin{array}{cc}
0 & 1 \\
\hline
000 & 011 \\
010 & 001
\end{array}
$$

$$
\begin{gathered}
((\neg A@0 \wedge \neg B@0 \wedge \neg C@0) \vee (\neg A@0 \wedge B@0 \wedge \neg C@0) \vee (A@0 \wedge B@0 \wedge C@0)) \\
\wedge \\
(\neg A@0 \wedge \neg A@1 \wedge (\neg B@0 \leftrightarrow B@1) \wedge (\neg C@0 \leftrightarrow C@1)) \\
= \\
\neg A@0 \wedge \neg A@1 \wedge ((\neg B@0 \wedge \neg C@0 \wedge B@1 \wedge C@1) \vee (B@0 \wedge \neg C@0 \wedge \neg B@1 \wedge C@1)
\end{gathered}
$$

Formulas as Data Structure: Relations

What logical operation corresponds to projection?

$$\Pi_1 \left(\begin{array}{cc} 0 & 1 \\ \hline 000 & 011 \\ 010 & 001 \end{array} \right) = \begin{array}{c} 1 \\ \hline 011 \\ 001 \end{array}$$

From $\neg A@0 \wedge \neg A@1 \wedge ((\neg B@0 \wedge \neg C@0 \wedge B@1 \wedge C@1) \vee (B@0 \wedge \neg C@0 \wedge \neg B@1 \wedge C@1))$ produce $(\neg A@1 \wedge B@1 \wedge C@1) \vee (\neg A@1 \wedge \neg B@1 \wedge C@1)$.

Existential and Universal Abstraction

Definition

Existential abstraction of ϕ with respect to x:

$$\exists x.\phi = \phi[\top/x] \vee \phi[\bot/x].$$

(Cf. Shannon expansion $\phi \equiv (x \wedge \phi[\top/x]) \vee (\neg x \wedge \phi[\bot/x])$)

Definition

Universal abstraction of ϕ with respect to x:

$$\forall x.\phi = \phi[\top/x] \wedge \phi[\bot/x].$$

∃-Abstraction

Example

$$\begin{aligned}
&\exists B.((A \to B) \land (B \to C))\\
&= ((A \to \top) \land (\top \to C)) \lor ((A \to \bot) \land (\bot \to C))\\
&\equiv C \lor \neg A\\
&\equiv A \to C
\end{aligned}$$

$$\begin{aligned}
&\exists AB.(A \lor B) = \exists B.(\top \lor B) \lor (\bot \lor B)\\
&= ((\top \lor \top) \lor (\bot \lor \top)) \lor ((\top \lor \bot) \lor (\bot \lor \bot))\\
&\equiv (\top \lor \top) \lor (\top \lor \bot) \equiv \top
\end{aligned}$$

Properties of Existential Abstraction

Theorem

Let ϕ be any formula over atomic propositions X and $v : X \to \{0,1\}$ any valuation of X.

1. *If $v(\phi) = 1$, then also $v(\exists x.\phi) = 1$.*
2. *If $v(\exists x.\phi) = 1$, then there is a valuation v' such that $v'(\phi) = 1$, and $v(y) = v'(y)$ for all $y \in X \backslash \{x\}$.*

$\forall$ and $\exists$-Abstraction with Truth-Tables

$\forall c$ and $\exists c$ eliminate the column for c by combining lines with the same valuation for variables other than c.

Example

$\exists c.(a \vee (b \wedge c)) \equiv a \vee b$

a	b	c	$a \vee (b \wedge c)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

a	b	$\exists c.(a \vee (b \wedge c))$
0	0	0
0	1	1
1	0	1
1	1	1

$\forall c.(a \vee (b \wedge c)) \equiv a$

a	b	$\forall c.(a \vee (b \wedge c))$
0	0	0
0	1	0
1	0	1
1	1	1

Example

From $\neg A@0 \wedge \neg A@1 \wedge ((\neg B@0 \wedge \neg C@0 \wedge B@1 \wedge C@1) \vee (B@0 \wedge \neg C@0 \wedge \neg B@1 \wedge C@1)$ produce $(\neg A@1 \wedge B@1 \wedge C@1) \vee (\neg A@1 \wedge \neg B@1 \wedge C@1)$.

$$\Phi = \neg A@0 \wedge \neg A@1 \wedge ((\neg B@0 \wedge \neg C@0 \wedge B@1 \wedge C@1) \vee (B@0 \wedge \neg C@0 \wedge \neg B@1 \wedge C@1)$$

$$\begin{aligned}
&\exists A@0 B@0 C@0.\Phi \\
&= \exists B@0 C@0.(\Phi[\bot/A@0] \vee \Phi[\top/A@0]) \\
&= \exists B@0 C@0.(\neg A@1 \wedge ((\neg B@0 \wedge \neg C@0 \wedge B@1 \wedge C@1) \vee (B@0 \wedge \neg C@0 \wedge \neg B@1 \wedge C@1)) \\
&= \exists C@0.((\neg A@1 \wedge (\neg C@0 \wedge B@1 \wedge C@1)) \vee (\neg A@1 \wedge ((\neg C@0 \wedge \neg B@1 \wedge C@1)))) \\
&= (\neg A@1 \wedge B@1 \wedge C@1) \vee (\neg A@1 \wedge \neg B@1 \wedge C@1)
\end{aligned}$$

Computing the Successors of a State Set

Procedure

INPUT:

- ϕ representing a set of states
- Θ_{01} formula for a relation

1. Compute the formula $\exists X_0.(\phi@0 \wedge \Theta_{01})$ where X_0 is all state variables with subscript 0 added.
2. Replace all remaining subscripts 1 by 0.

Denote the resulting formula by $img_{\Theta_{01}}(\phi)$.

Computing All Reachable States

1. $i := 0$
2. $\Phi_0 := I@0$ (The initial states as a formula)
3. $i := i + 1$
4. $\Phi_i := \Phi_{i-1} \vee img_{\Theta_{01}}(\Phi_{i-1})$
5. if $\Phi_i \not\models \Phi_{i-1}$ then go to step 3

Φ_i represents the set of all reachable states

SAT-based Reachability vs. Symbolic Breadth-First

Theorem

$$I@0 \wedge \Theta_{01} \wedge \cdots \wedge \Theta_{(T-1)T} \wedge G@T$$

is satisfiable iff the following is:

$$\left(\exists X_0 \cup \cdots X_{T-1}(I@0 \wedge \Theta_{01} \wedge \cdots \wedge \Theta_{(T-1)T})\right) \wedge G@T$$

Stochastic Actions

- What to do when actions are stochastic (non-deterministic)?
- Multiple possible successor states
- Reaching goals cannot always be guaranteed
- Options:
  1. Try to maximize probability of reaching goals
  2. Try to minimize expected cost of reaching goals
  3. Try to maximize expected rewards (no goal states!)
- This lecture: Markov decision processes (option 3)

Markov Decision Processes (MDP)

Definition (MDP $\langle S, A, P, R \rangle$)

- S is a (finite) set of states
- A is a (finite) set of actions
- $P : S \times A \times S \to \mathbb{R}$ gives transition probabilities
- $R : S \times A \times S \to \mathbb{R}$ is a reward function

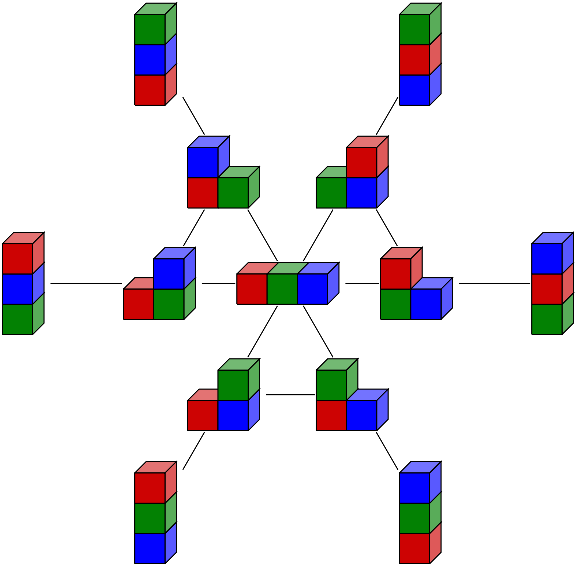
Notice that

- Plan/policy given as $\pi : S \to A$
- Usually no designated initial state
- Reward functions are often $R(s, a) : S \times A \to \mathbb{R}$

Policies for MDPs (deterministic example)

All actions

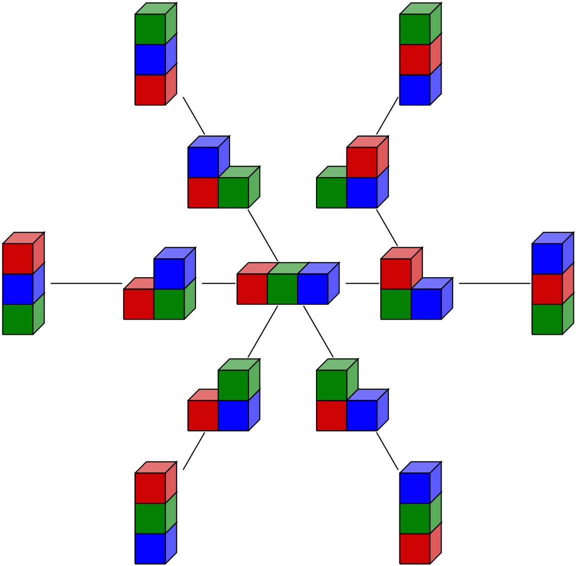
Policy/plan



Policies for MDPs (deterministic example)

All actions

Policy/plan



Value of an Action Sequence

1. finite sum
$$\sum_{i=0}^{n} R(s_i, a_i, s_{i+1})$$
2. geometrically discounted sum (with $0 < \gamma < 1$)
$$\sum_{i=0}^{n} \gamma^i \cdot R(s_i, a_i, s_{i+1})$$
3. average
$$\lim_{N \to \infty} \frac{\sum_{i=0}^{N} R(s_i, a_i, s_{i+1})}{N}$$

Value of an Action Sequence

- Finite sums are used when
  - time horizon is bounded, or
  - approximate infinite with finite: receding-horizon control
- Discounted sums are used often
  - Finite sum for infinite sequences when $\gamma < 0$
  - Easy to handle in algorithms (the Bellman equation)
- Averages useful, but difficult to handle
  - Bellman equation does not apply
  - Easier in special cases only (unichain)

Choice of the Discount Factor γ

- γ close to 0: Emphasis on short-term rewards
- γ close to 1: Emphasis on long-term rewards

Example

	value with			
rewards	$\gamma = 0.1$	$\gamma = 0.8$	$\gamma = 0.9$	$\gamma = 0.99$
5 0 0 0 20	5.002	13.192	18.122	24.212
20 0 0 0 5	20.00005	22.048	23.281	24.803

Bellman equation

The value of state s under the best possible plan/policy given by the Bellman equation

$$v(s) = \max_{a \in A} \sum_{s' \in S} P(s, a, s')[R(s, a, s') + \gamma v(s')]$$

Algorithms for Finding Optimal Policies

Value Iteration

- Iterate by finding value functions closer to optimal
- Policy implicit in value function
- Terminate when change smaller than given bound

Policy Iteration

- Iterate by improving policy bit by bit
- Fewer rounds than Value Iteration
- Termination when policy not improved

Value Iteration

1. Let $n := 0$ and $v_0 : S \to \mathbb{R}$ be any value function.
2. For every $s \in S$

$$v_{n+1}(s) = \max_{a \in A} \left(\sum_{s' \in S} P(s, a, s') \left(R(s, a, s') + \gamma v_n(s') \right) \right).$$

   Go to 3 if $|v_{n+1}(s) - v_n(s)| < \frac{\epsilon(1-\gamma)}{2\gamma}$ for all $s \in S$.
   Otherwise set $n := n + 1$ and repeat this step.
3. Policy $\pi : S \to A$ given by

$$\pi(s) = \arg\max_{a \in A} \sum_{s' \in S} P(s, a, s') \left(R(s, a, s') + \gamma v_n(s') \right).$$

Value Iteration

Theorem

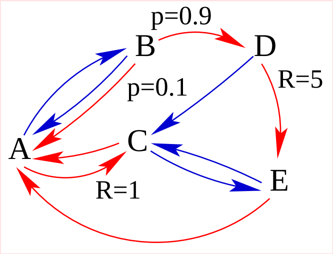
Let v_π be the value function of the policy produced by the value iteration algorithm, and let v^* be the value function of an optimal policy. Then $|v^*(s) - v_\pi(s)| \leq \epsilon$ for all $s \in S$.

Value Iteration

Example

Let $\gamma = 0.6$.

i	$v_i(A)$	$v_i(B)$	$v_i(C)$	$v_i(D)$	$v_i(E)$
0	0.000	0.000	0.000	0.000	0.000
1	1.000	0.000	0.000	5.000	0.000
2	1.000	2.760	0.600	5.000	0.600
3	1.656	2.760	0.600	5.360	0.600
4	1.656	2.994	0.994	5.360	0.994
5	1.796	2.994	0.994	5.596	0.994
6	1.796	3.130	1.078	5.596	1.078
7	1.878	3.130	1.078	5.647	1.078
8	1.878	3.162	1.127	5.647	1.127
⋮					
19	1.912	3.186	1.147	5.688	1.147
20	1.912	3.186	1.147	5.688	1.147



Policy Iteration

- Policy Iteration finds optimal policies.
- Slightly more complicated to implement than Value Iteration: on each iteration
  - the value of the current policy is evaluated, and
  - the current policy is improved if possible.
- Fewer iterations than with Value Iteration.
- Value Iteration in practice usually more efficient.

Policy Evaluation with Linear Equations

Given a policy π, its value v_π with discount constant γ satisfies for every $s \in S$

$$v_\pi(s) = \sum_{s' \in S} P(s, \pi(s), s') \left(R(s, \pi(s), s') + \gamma v_\pi(s') \right)$$

This yields a system of $|S|$ linear equations and $|S|$ unknowns. The solution of these equations gives the value of the policy in each state.

Policy Evaluation with Linear Equations

Example

Consider the policy

$$\pi(A) = R, \pi(B) = R, \pi(C) = B, \pi(D) = R, \pi(E) = B$$

$$\begin{aligned}
v_\pi(A) &= R(A, R) + 0\gamma v_\pi(A) + 0\gamma v_\pi(B) + 1\gamma v_\pi(C) + 0\gamma v_\pi(D) + 0\gamma v_\pi(E) \\
v_\pi(B) &= R(B, R) + 0.1\gamma v_\pi(A) + 0\gamma v_\pi(B) + 0\gamma v_\pi(C) + 0.9\gamma v_\pi(D) + 0\gamma v_\pi(E) \\
v_\pi(C) &= R(C, B) + 0\gamma v_\pi(A) + 0\gamma v_\pi(B) + 0\gamma v_\pi(C) + 0\gamma v_\pi(D) + 1\gamma v_\pi(E) \\
v_\pi(D) &= R(D, R) + 0\gamma v_\pi(A) + 0\gamma v_\pi(B) + 0\gamma v_\pi(C) + 0\gamma v_\pi(D) + 1\gamma v_\pi(E) \\
v_\pi(E) &= R(E, B) + 0\gamma v_\pi(A) + 0\gamma v_\pi(B) + 1\gamma v_\pi(C) + 0\gamma v_\pi(D) + 0\gamma v_\pi(E)
\end{aligned}$$

$$\begin{aligned}
v_\pi(A) &= 1 && +\gamma v_\pi(C) \\
v_\pi(B) &= 0 + 0.1\gamma v_\pi(A) && +0.9\gamma v_\pi(D) \\
v_\pi(C) &= 0 && +\gamma v_\pi(E) \\
v_\pi(D) &= 5 && +\gamma v_\pi(E) \\
v_\pi(E) &= 0 && +\gamma v_\pi(C)
\end{aligned}$$

Policy Evaluation with Linear Equations

$$
\begin{array}{rrrrrcl}
v_\pi(A) & & -\gamma v_\pi(C) & & & = & 1 \\
-0.1\gamma v_\pi(A) & +v_\pi(B) & & -0.9\gamma v_\pi(D) & & = & 0 \\
 & & v_\pi(C) & & -\gamma v_\pi(E) & = & 0 \\
 & & & v_\pi(D) & -\gamma v_\pi(E) & = & 5 \\
 & & -\gamma v_\pi(C) & & +v_\pi(E) & = & 0
\end{array}
$$

Solving with $\gamma = 0.5$ we get

$$
\begin{array}{rrrrrcl}
v_\pi(A) & & & & & = & 1 \\
 & v_\pi(B) & & & & = & 2.3 \\
 & & v_\pi(C) & & & = & 0 \\
 & & & v_\pi(D) & & = & 5 \\
 & & & & v_\pi(E) & = & 0
\end{array}
$$

This is the value function of the policy.

Policy Iteration

1. $n := 0$
2. Let $\pi_0 : S \to A$ be any mapping from states to actions.
3. Compute $v_{\pi_n}(s)$ for all $s \in S$.
4. For all $s \in S$

$$\pi_{n+1}(s) = \arg\max_{a \in A} \left(\sum_{s' \in S} P(s, a, s')(R(s, a, s') + \gamma v_{\pi_n}(s') \right)$$

5. $n := n + 1$
6. If $n = 1$ or $v_{\pi_n} \neq v_{\pi_{n-1}}$ then go to 3.

Policy Iteration

Theorem

If the number of states is finite, then Policy Iteration terminates after a finite number of steps and returns an optimal policy.

Proof idea.

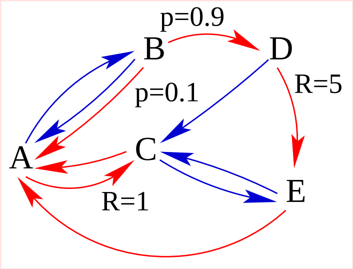
There is only a finite number of different policies, and at each step a properly better policy is found or the algorithm terminates. □

The number of iterations needed for finding an ϵ-optimal policy by policy iteration is never higher than the number of iterations needed by value iteration.

Policy Iteration

Example

itr.	$\pi(A)$	$\pi(B)$	$\pi(C)$	$\pi(D)$	$\pi(E)$	$v_\pi(A)$	$v_\pi(B)$	$v_\pi(C)$	$v_\pi(D)$	$v_\pi(E)$
1	R	R	R	R	R	1.56	3.09	0.93	5.56	0.93
2	B	R	R	R	R	1.91	3.18	1.14	5.68	1.14



MDPs with Very Large State Spaces

- Both Value Iteration and Policy Iteration “visit” the whole state space
- These algorithms not feasible beyond 10^7 states
- Heuristic search algorithms for solving MDPs:
  - Not all states need to be visited
  - heuristics help focusing search
  - LAO*, LRTDP, ...
- Symbolic data structures (Algebraic Decision Diagrams (ADD), other generalizations of BDD)

Reinforcement Learning

What if system model is incomplete?

- reward function $R(s, a, s')$ is unknown
- transition probabilities $P(s, a, s')$ unknown

Find near-optimal policies by Reinforcement Learning:

- Learning and execution are interleaved
- With every new reward and state, update model

Reinforcement Learning

- Applications:
  - robotics
  - control of distributed systems: power, telecom, ...
  - game playing
- Lots of different algorithms and approaches
- Issues:
  - Size of the state space
  - Slow learning when lots of states
- This lecture: brief intro to Q-learning

Q-Learning

What is given:

- action set A,
- state set S,
- discount factor γ (as with MDPs)
- learning rate λ (higher $\rightarrow$ faster learning)

What is learned: Q-values $Q(s, a) : S \times A \rightarrow \mathbb{R}$ which

- estimate the value of taking a in s
- summarize both
    - the transition probabilities from s with a, and
    - the values of successors of s with a

Q-Learning

1. Let $Q(s, a) = 0$ for all $s \in S$ and $a \in A$
2. $s :=$ current state in the beginning
3. Choose action a based on $Q(s, a)$ (*see next slide*)
4. Execute a to obtain new state s' and reward r
5. $Q(s, a) := (1-\lambda)Q(s, a)+\lambda(r+\gamma\cdot\max_{a\in A} Q(s', a))$
6. Set $s := s'$ and go to 3

Step 3 tries to balance between

- exploration: Improving accuracy of $Q(s, a)$
- exploitation: Taking action a with highest $Q(s, a)$

Exploration vs. Exploitation

Choice of action based on $Q(s, a_1), \ldots, Q(s, a_n)$:

- Prefer actions a with high $Q(s, a)$ (exploitation)
- Try also other actions (exploration)
- Best to base this on an estimate on confidence
  - How much confidence on current $Q(s, a)$?
  - How many times has a been tried before in s?
- More exploration early
- More exploitation later
- Lots of different alternatives how to do this!