Artificial Intelligence

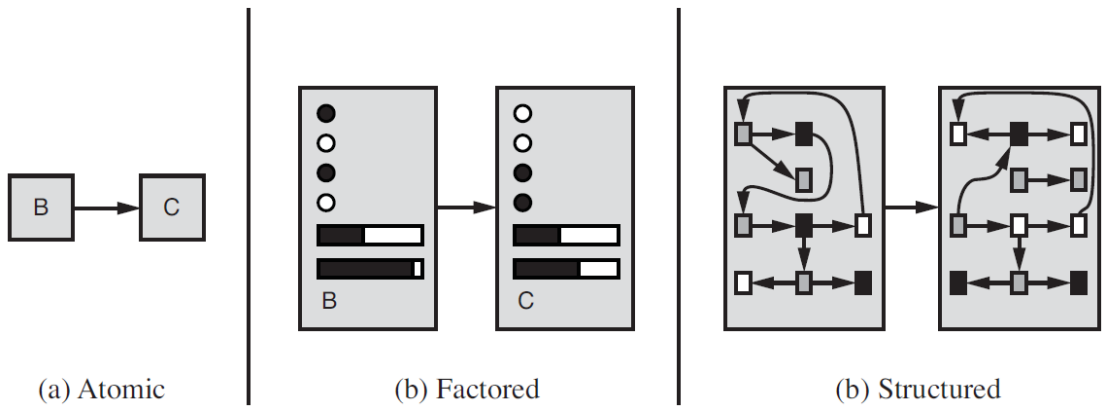
Other models of interactive domains

Marc Toussaint
University of Stuttgart
Winter 2018/19

Basic Taxonomy of domain models

Taxonomy of domains I

- Domains (or models of domains) can be distinguished based on their state representation
  - Discrete, continuous, hybrid
  - Factored
  - Structured/relational



Relational representations of state

- The world is composed of objects; its state described in terms of properties and relations of objects. Formally
  - A set of constants (referring to objects)
  - A set of predicates (referring to object properties or relations)
  - A set of functions (mapping to constants)

- A (grounded) state can then described by a conjunction of predicates (and functions). For example:
  - Constants: $C_1, C_2, P_1, P_2, SFO, JFK$
  - Predicates: $At(.,.), Cargo(.), Plane(.), Airport(.)$
  - A state description:
    $At(C_1, SFO) \wedge At(C_2, JFK) \wedge At(P_1, SFO) \wedge At(P_2, JFK) \wedge Cargo(C_1) \wedge$
    $Cargo(C_2) \wedge Plane(P_1) \wedge Plane(P_2) \wedge Airport(JFK) \wedge Airport(SFO)$

Taxonomy of domains II

- Domains (or models of domains) can additionally be distinguished based on:

- Categories of Russel & Norvig:
  - Fully observable vs. partially observable
  - Single agent vs. multiagent
  - Deterministic vs. stochastic
  - Known vs. unknown
  - Episodic vs. sequential
  - Static vs. dynamic
  - Discrete vs. continuous

- We add:
  - Time discrete vs. time continuous

Overview of common domain models

	prob.	rel.	multi	PO	cont.time
table	-	-	-	-	-
PDDL (STRIPS rules)	-	+	+	-	-
NDRs	+	+	-	-	-
MDP	+	-	-	-	-
relational MDP	+	+	-	-	-
POMDP	+	-	-	+	-
DEC-POMDP	+	-	+	+	-
Games	-	+	+	-	-
differential eqns. (control)	-	-		+	+
stochastic diff. eqns. (SOC)	+	-		+	+

PDDL: Planning Domain Definition Language, STRIPS: STanford Research Institute Problem Solver, NDRs: Noisy Deictic Rules, MDP: Markov Decision Process, POMDP: Partially Observable MDP, DEC-POMDP: Decentralized POMDP, SOC: Stochastic Optimal Control

PDDL

- Planning Domain Definition Language
  Developed for the 1998/2000 International Planning Competition (IPC)

$$
\begin{aligned}
&Init(At(C_1, SFO) \land At(C_2, JFK) \land At(P_1, SFO) \land At(P_2, JFK)\\
&\quad \land Cargo(C_1) \land Cargo(C_2) \land Plane(P_1) \land Plane(P_2)\\
&\quad \land Airport(JFK) \land Airport(SFO))\\
&Goal(At(C_1, JFK) \land At(C_2, SFO))\\
&Action(Load(c, p, a),\\
&\quad \text{PRECOND: } At(c, a) \land At(p, a) \land Cargo(c) \land Plane(p) \land Airport(a)\\
&\quad \text{EFFECT: } \neg At(c, a) \land In(c, p))\\
&Action(Unload(c, p, a),\\
&\quad \text{PRECOND: } In(c, p) \land At(p, a) \land Cargo(c) \land Plane(p) \land Airport(a)\\
&\quad \text{EFFECT: } At(c, a) \land \neg In(c, p))\\
&Action(Fly(p, from, to),\\
&\quad \text{PRECOND: } At(p, from) \land Plane(p) \land Airport(from) \land Airport(to)\\
&\quad \text{EFFECT: } \neg At(p, from) \land At(p, to))
\end{aligned}
$$

Figure 10.1 A PDDL description of an air cargo transportation planning problem.

(from Russel & Norvig)

- PDDL describes a deterministic mapping $(s, a) \mapsto s'$, but
  - using a set of action schema (rules) of the form
    ActionName(...) : PRECONDITION $\rightarrow$ EFFECT
  - where action arguments are variables and the preconditions and effects are conjunctions of predicates

PDDL

$Init(On(A, Table) \land On(B, Table) \land On(C, A)$
$\quad \land Block(A) \land Block(B) \land Block(C) \land Clear(B) \land Clear(C))$
$Goal(On(A, B) \land On(B, C))$
$Action(Move(b, x, y),$
$\quad$ PRECOND: $On(b, x) \land Clear(b) \land Clear(y) \land Block(b) \land Block(y) \land$
$\quad\quad (b \neq x) \land (b \neq y) \land (x \neq y),$
$\quad$ EFFECT: $On(b, y) \land Clear(x) \land \neg On(b, x) \land \neg Clear(y))$
$Action(MoveToTable(b, x),$
$\quad$ PRECOND: $On(b, x) \land Clear(b) \land Block(b) \land (b \neq x),$
$\quad$ EFFECT: $On(b, Table) \land Clear(x) \land \neg On(b, x))$

Figure 10.3 A planning problem in the blocks world: building a three-block tower. One solution is the sequence $[MoveToTable(C, A), Move(B, Table, C), Move(A, Table, B)]$.

Figure 10.4 Diagram of the blocks-world problem in Figure 10.3.

PDDL

- The state-of-the-art solvers are actually A*methods. But the heuristics!!
- Scale to huge domains
- Fast-Downward is great

Noisy Deictic Rules (NDRs)

- Noisy Deictic Rules (Pasula, Zettlemoyer, & Kaelbling, 2007)
- A probabilistic extension of "PDDL rules":

$$grab(X): \quad on(X,Y),\ ball(X),\ cube(Y),\ table(Z)$$
$$\rightarrow \quad \begin{cases} 0.7 & : & inhand(X),\ \neg on(X,Y) \\ 0.2 & : & on(X,Z),\ \neg on(X,Y) \\ 0.1 & : & \text{noise} \end{cases}$$

- These rules define a *probabilistic transition probability*

$$P(s'|s,a)$$

Namely, if (s,a) has a unique covering rule r, then

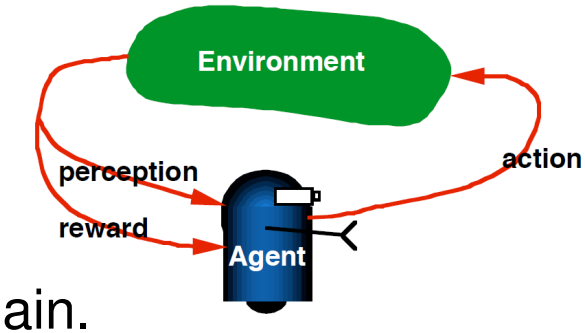
$$P(s'|s,a) = P(s'|s,r) = \sum_{i=0}^{m_r} p_{r,i}\ P(s'|\Omega_{r,i},s)$$

where $P(s'|\Omega_{r,i},s)$ describes the deterministic state transition of the ith outcome (see Lang & Toussaint, JAIR 2010).

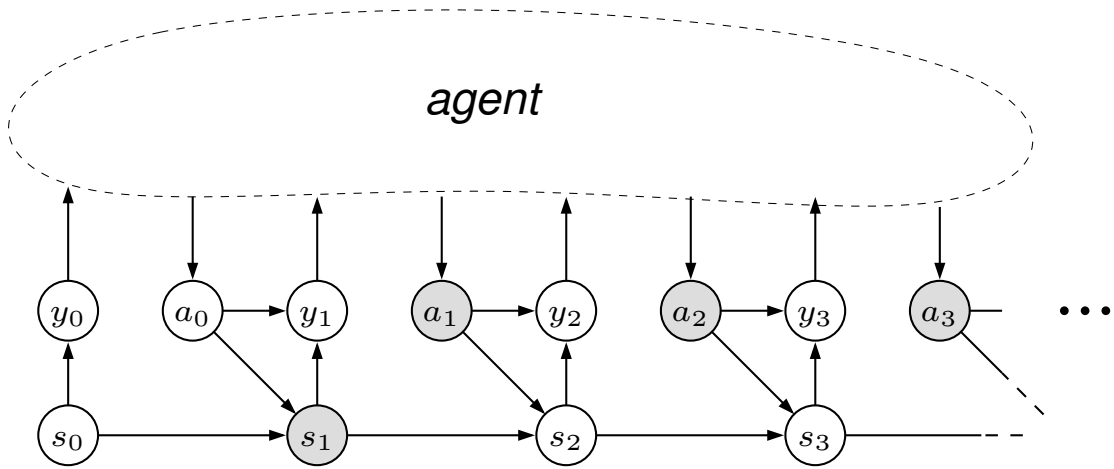
- While such rule based domain models originated from classical AI research, the following were strongly influenced also from stochastics, decision theory, Machine Learning, etc..

Partially Observable MDPs

Recall the general setup



- We assume the agent is in *interaction* with a domain.
  - The world is in a state $s_t \in \mathcal{S}$
  - The agent senses observations $y_t \in \mathcal{O}$
  - The agent decides on an action $a_t \in \mathcal{A}$
  - The world transitions in a new state s_{t+1}

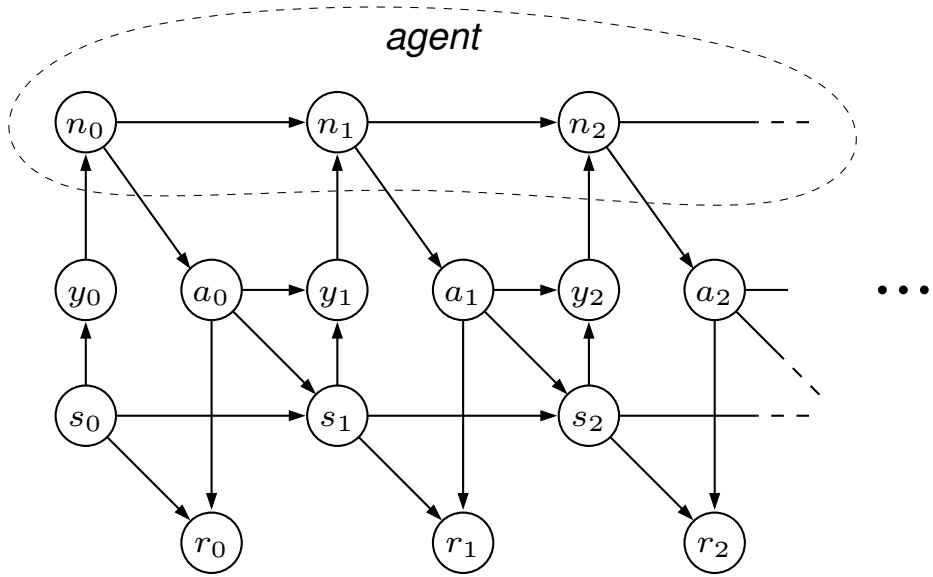


- Generally, an agent maps the history to an action,
  $h_t = (y_{0:t}, a_{0:t\text{-}1}) \mapsto a_t$

POMDPs

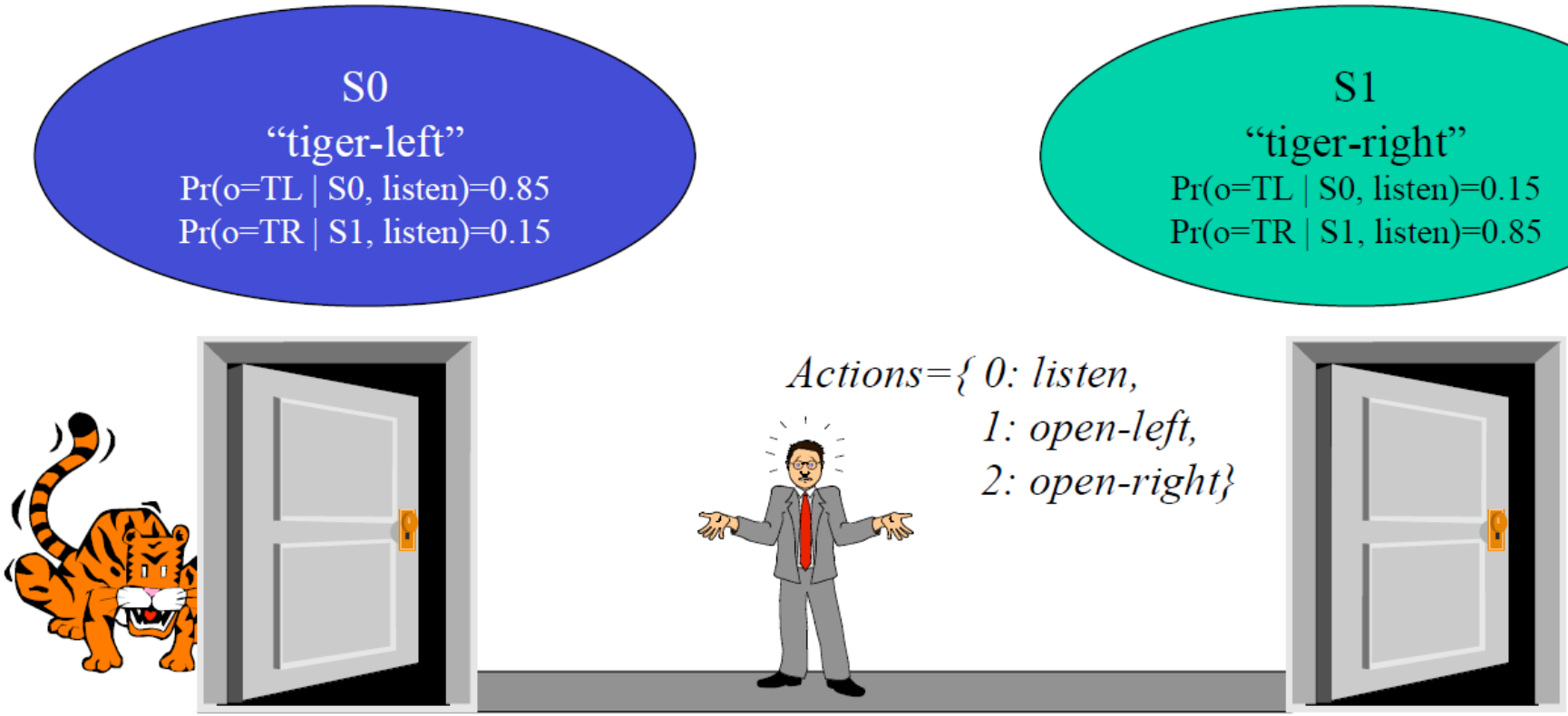
- Partial observability adds a totally new level of complexity!
- Basic alternative agent models:
  - The agent maps $y_t \mapsto a_t$
    (stimulus-response mapping.. non-optimal)
  - The agent stores all previous observations and maps $y_{0:t}, a_{0:t\text{-}1} \mapsto a_t$
    (ok)
  - The agent stores only the recent history and maps $y_{t-k:t}, a_{t-k:t\text{-}1} \mapsto a_t$
    (crude, but may be a good heuristic)
  - The agent is some machine with its own **internal state** n_t, e.g., a computer, a finite state machine, a brain... The agent maps $(n_{t\text{-}1}, y_t) \mapsto n_t$ (internal state update) and $n_t \mapsto a_t$
  - The agent maintains a full probability distribution (**belief**) $b_t(s_t)$ over the state, maps $(b_{t\text{-}1}, y_t) \mapsto b_t$ (Bayesian belief update), and $b_t \mapsto a_t$

POMDP coupled to a state machine agent



http://www.darpa.mil/grandchallenge/index.asp

- The tiger problem: a typical POMDP example:



Reward Function

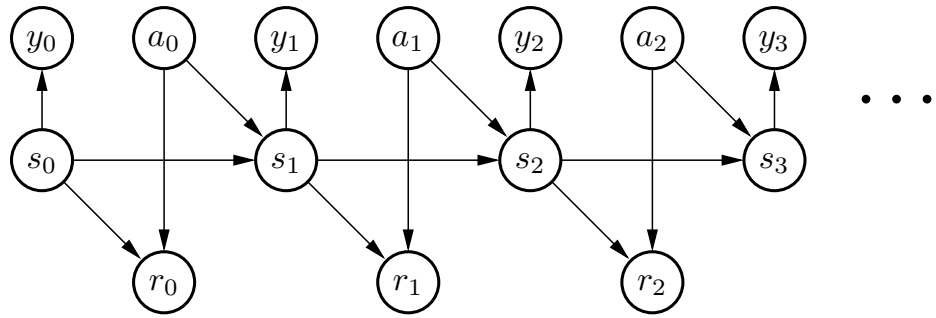
- *Penalty for wrong opening: -100*
- *Reward for correct opening: +10*
- *Cost for listening action: -1*

Observations

- *to hear the tiger on the left (TL)*
- *to hear the tiger on the right(TR)*

(from the a "POMDP tutorial")

Solving POMDPs via Dynamic Programming in Belief Space



- Again, the value function is a function over the belief

$$V(b) = \max_a \left[R(b,s) + \gamma \sum_{b'} P(b'|a,b)\, V(b') \right]$$

- Sondik 1971: V is piece-wise linear and convex: Can be described by m vectors $(\alpha_1, .., \alpha_m)$, each $\alpha_i = \alpha_i(s)$ is a function over discrete s

$$V(b) = \max_i \sum_s \alpha_i(s) b(s)$$

Exact dynamic programming possible, see Pineau et al., 2003

Approximations & Heuristics

- Point-based Value Iteration (Pineau et al., 2003)
  – Compute $V(b)$ only for a finite set of belief points

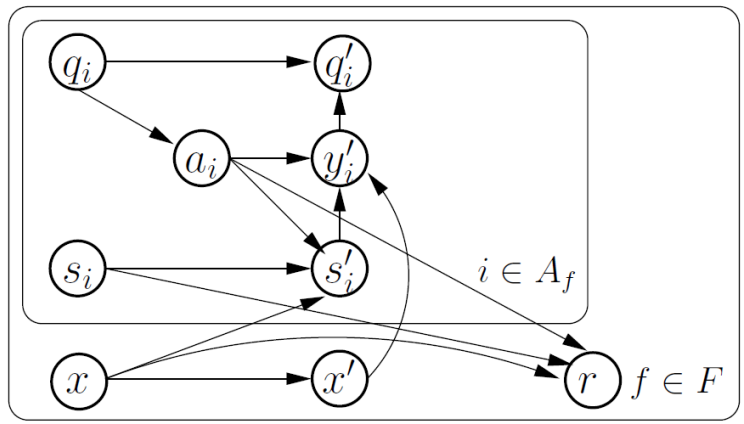
- Discard the idea of using belief to "aggregate" history
  – Policy directly maps history (window) to actions
  – Optimize finite state controllers (Meuleau et al. 1999, Toussaint et al. 2008)

Further reading

- *Point-based value iteration: An anytime algorithm for POMDPs.* Pineau, Gordon & Thrun, IJCAI 2003.
- The standard references on the "POMDP page"
  `http://www.cassandra.org/pomdp/`
- *Bounded finite state controllers.* Poupart & Boutilier, NIPS 2003.
- *Hierarchical POMDP Controller Optimization by Likelihood Maximization.* Toussaint, Charlin & Poupart, UAI 2008.

Decentralized POMDPs

- Finally going multi agent!



(from Kumar et al., IJCAI 2011)

- This is a special type (simplification) of a general DEC-POMDP

- Generally, this level of description is very general, but NEXP-hard
  Approximate methods can yield very good results, though

Controlled System

- Time is continuous, $t \in \mathbb{R}$
- The system state, actions and observations are continuous, $x(t) \in \mathbb{R}^n, u(t) \in \mathbb{R}^d, y(t) \in \mathbb{R}^m$

- A controlled system can be described as

  linear:

  $$\dot{x} = Ax + Bu$$
  $$y = Cx + Du$$

  with matrices A, B, C, D

  non-linear:

  $$\dot{x} = f(x, u)$$
  $$y = h(x, u)$$

  with functions f, h

- A typical "agent model" is a *feedback regulator* (stimulus-response)

$$u = Ky$$

Stochastic Control

- The differential equations become stochastic

$$dx = f(x, u)\ dt + d\xi_x$$
$$dy = h(x, u)\ dt + d\xi_y$$

  $d\xi$ is a Wiener processes with $\langle d\xi, d\xi \rangle = C_{ij}(x, u)$

- This is the control theory analogue to POMDPs

Overview of common domain models

	prob.	rel.	multi	PO	cont.time
table	-	-	-	-	-
PDDL (STRIPS rules)	-	+	+	-	-
NID rules	+	+	-	-	-
MDP	+	-	-	-	-
relational MDP	+	+	-	-	-
POMDP	+	-	-	+	-
DEC-POMDP	+	-	+	+	-
Games	-	+	+	-	-
differential eqns. (control)	-	-		+	+
stochastic diff. eqns. (SOC)	+	-		+	+

PDDL: Planning Domain Definition Language, STRIPS: STanford Research Institute Problem Solver, NDRs: Noisy Deictic Rules, MDP: Markov Decision Process, POMDP: Partially Observable MDP, DEC-POMDP: Decentralized POMDP, SOC: Stochastic Optimal Control