CSE 571 - FALL 99. HANDOUT 1. VERSION 0.1

Chitta Baral

Department of Computer Science and Engineering
Arizona State University
Tempe, AZ 85287-5406 USA
chitta@asu.edu

http://cs.utep.edu/chitta/

August 30, 1999

INTRODUCTION TO THE COURSE

My biased big picture of AI

- Deliberative vs Reactive
- What is the connection between them?

Overview of the course

- AI consists of many subtopics: Search, Knowledge representation, learning, natural language processing, vision, robotics, planning, etc.
- Often they appear disconnected
- We will focus on many of these with a common running theme: planning
- No! This course is not about planning.
- We will explore many subfields of AI.
- But to tie them together we will use a common running theme, instead of using different examples for different topics.

The course content

- Languages
- Search
- Constraint satisfaction
- Knowledge representation, and logical langauges
- Autonomous agents
- Uncertainity
- Learning
- Robotics

LANGUAGES

Property of generic langauges

- The syntax.

  What are valid sentences in this language.

- The Semantics.

  What does a partcular sentence 'mean'?

  What do a set of sentences 'mean'?

Languages in AI

- Instances.
  - Planning and Search: Language to describe the state space
  - Reasoning: Language to describe knowledge
  - Diagnosis: Language to describe a system
- Common property

  Background Knowledge + Specific knowledge $\models$ Query
- Examples
  - State space description + Initial state description $\models$ α is a plan that achieves goal G from the initial situation.
  - Knowledge $\models$ inference

    Normally birds fly, Tweety is a bird $\models$ Tweety flies.
  - System description + Observations $\models$ a diagnosis.

- Bayes' net + Probability of a set X of nodes $\models$ Conditional probability of a set of nodes Y given Z.

- Things to do
  - Syntax to express background knowledge
  - Syntax to express specific knowledge
  - Syntax to express queries
  - (They are not required to be same. Not required to be different either.)
  - Semantics: Defining $\models$
  - Implementation: Implementing $\models$

STRIPS – A language used in planning

- Syntax: $\langle \mathcal{L}_S, \mathcal{O}_s, \mathcal{I}_S, \mathcal{G}_S \rangle$
  - State Language ($\mathcal{L}_S$): constants (fixed symbols) and relational symbols (fluent symbols).
    Fluent atoms are of the form $p(t_1, \ldots, t_k)$, where p is a relational symbol and t_is are terms.
    Example: on(a,b)
  - Operators ($\mathcal{O}_S$): Each operator *op* has a precondition, an add and a delete list given by a set of atoms.
  - Initial situation ($\mathcal{I}_S$): A set of fluent atoms.
  - Goal situation ($\mathcal{G}_S$): A set of fluent atoms.
- Semantic Structure (Interpretations) : $\langle S, s_0, S_G, A, next \rangle$
  - S: a finite set of states.

- $s_0 \in S$: the initial state.
- $S_G \subseteq S$: a non-empty set of goal states.
- $A(s)$: Set of actions a applicable in state s.
- $next$: A transition function from action and states to states.
  i.e., For any state s, and action $a \in A(s)$, $next(a, s)$ is the state that will be reached by executing a in s.
- Solution (Plan) : A sequence of actions $a_0, \ldots, a_n$, such that $s_{i+1} = f(a_i, s_i)$, $a_i \in A(s_i)$ and $s_{n+1} \in S_G$.
  i.e., Interpretaon $\models$ a sequence of action is the plan.

• Semantics: From syntax to a precise interpretation (a model).
  Any STRIPS planning problem $P = \langle \mathcal{L}_S, \mathcal{O}_s, \mathcal{I}_S, \mathcal{G}_S \rangle$ is given a precise meaning by mapping it into the interpretation
  $S(P) = \langle S, s_0, S_G, A, next \rangle$ where
  - the states S are sets of atoms from $\mathcal{L}_S$
  - s_0 is $\mathcal{I}_S$

  - The goals states are states s such that $\mathcal{G}_S \subseteq s$
  - $A(s)$ is the subset of operators $op \in \mathcal{O}_S$ such that $Prec(op) \subseteq s$.
  - $next(a, s) = s \cup Add(a) \setminus Del(a)$, for $a \in A(s)$.
  - $P \models a_1, \ldots, a_n$ is a plan.

- Immediate extensions
  - Not just plan, but reason about actions.
  - negative literals in the add, del and precond lists.
  - conditional effects

- Further extensions
  - Concurrent actions
  - Classical and causal constraints
  - Functional fluents

The language $\mathcal{A}$: Gelfond and Lifschitz

- A high level language
  - for expressing effects of actions,
  - and hypothetically reason about values of fluents after executing a sequence of actions;
  - which is simple, and intuitive,
  - but yet has a precise semantics;
  - unlike a natural language description (Yale shooting problem).
- It extends and enhances STRIPS by
  - allowing negative literals in the executability conditions
  - allowing conditional effects with both positive and negative conditions
  - allowing initial states to be incomplete

- The transition diagram based semantics is easier to describe, and seems to be less error prone than direct logical formulation.
- A logical formulation can now be precisely evaluated, in terms of whether it is correct or not.

Syntax of $\mathcal{A}$

- Notation: actions (a_i); fluents (f_i, p_i); fluent literals ($f_i, \neg f_i, p_i, \neg p_i$); positive and negative fluent literals.
- Action Description:
  - a **causes** f **if** $p_1, \ldots, p_n$ — *load* **causes** *loaded*; *shoot* **causes** $\neg$*alive* **if** *loaded*
  - **executable** a **if** $p_1, \ldots, p_n$ — **executable** *load* **if** $\neg$*loaded*
- Facts about the initial state:

  initially f — **initially** $\neg$*loaded*; **initially** *alive*
- Queries:

  f **after** $a_1, \ldots, a_n$ — $\neg$*alive* **after** *load*; *shoot*

Semantics of $\mathcal{A}$

- states (σ)
- f holds in σ iff $f \in \sigma$. $\neg f$ holds in σ iff $f \notin \sigma$.
- transition function (Φ) : actions $\times$ states $\rightarrow$ states.
- Domain description (D) : action description + facts about the initial state
- $E^+(a, \sigma) = \{f$: there exists an axiom a **causes** f **if** $p_1, \ldots, p_n$ in D such that $p_1, \ldots, p_n$ hold in $\sigma\}$
- $E^-(a, \sigma) = \{f$: there exists an axiom a **causes** $\neg f$ **if** $p_1, \ldots, p_n$ in D such that $p_1, \ldots, p_n$ hold in $\sigma\}$
- a is executable in σ if $E^+(a, \sigma) \cap E^-(a, \sigma) = \emptyset$ and there exists a dynamic causal rule **executable** a **if** $p_1, \ldots, p_n \in D$ such that $p_1, \ldots, p_n$ hold in σ.

- Φ is a transition function for a domain description D, if for all actions a, and states σ, if a is executable in σ then:
  $\Phi(a, \sigma) = (\sigma \cup E^{+}(a, \sigma)) \setminus E^{-}(a, \sigma)$
  Otherwise $\Phi(a, \sigma)$ is undefined.
- (Φ, σ) is a model of D if Φ is a transition function of D and all facts about the initial state in D hold in σ.
- A query f **after** $a_1, \ldots, a_n$ is said to be true in a model (Φ, σ) of D if f holds in $\Phi(a_n, \ldots \Phi(a_1, \sigma) \ldots)$
- $D \models f$ **after** $a_1, \ldots, a_n$ iff for all models (Φ, σ) of D, f **after** $a_1, \ldots, a_n$ is true in (Φ, σ).

Example 1

- The domain description
  - *load* **causes** *loaded*
  - *shoot* **causes** $\neg$*alive* **if** *loaded*
  - **executable** *load* **if** $\neg$*loaded*
  - **initially** $\neg$*loaded*
  - **initially** *alive*
- What are the models of the above description?
- What are some of the conclusions that we can make? Why?

Example 2: Incomplete information about initial situation

- The domain description
  - *load* **causes** *loaded*
  - *shoot* **causes** $\neg$*alive* **if** *loaded*
  - **executable** *load* **if** $\neg$*loaded*
  - **executable** *load* **if** $\neg$*loaded*
  - No idea if gun is initially loaded. (No **initially** $\neg$*loaded*)
  - **initially** *alive*
- What are the models of the above description?
- What are some of the conclusions that we can make? Why?

  Ex: Is *alive* **after** *shoot*?

$\mathcal{A}$: Incorporating hypothesis

- Hypothesis: f **after** $a_1, \ldots, a_m$
  If $m = 0$, we say **currently** f.
- Hypothesis vs observations.
- Hyp : A set of hypothesis.
- $D \models_{Hyp} f$ **after** $a_1, \ldots, a_n$ iff for all models (Φ, σ) of D where all the statements in Hyp are true, f **after** $a_1, \ldots, a_n$ is also true.
- An example
  - *load* **causes** *loaded*
  - *shoot* **causes** $\neg alive$ **if** *loaded*
  - **executable** *load* **if** $\neg loaded$
  - **initially** *alive*
  - Hyp: $\{alive$ **after** $shoot\}$

– $D \models_{Hyp}$ **currently** $\neg loaded$

Motivation; Syntax of $\mathcal{A}_C$

- Simple Inheritance
  - $lift$ **causes** $lifted$ Mary lifting a bowl
  - $open$ **causes** $opened$ John opening a door
  - $opened$ **after** $\{lift, open\}$
  - $lifted$ **after** $\{lift, open\}$
- Overriding inheritance
  - l_lift **causes** $spilled$
  - r_lift **causes** $spilled$
  - $\{l\_lift, r\_lift\}$ **causes** $\neg spilled$ **if** $\neg spilled$
  - **initially** $\neg spilled$
  - $spilled$ **after** l_lift $spilled$ **after** r_lift
  - $\neg spilled$ **after** $\{l\_lift, r\_lift\}$

Semantics of $\mathcal{A}_C$

- Execution of a in σ *immediately causes* f if there is an axiom a **causes** f **if** $p_1, \ldots, p_n$ in D such that $p_1, \ldots, p_n$ hold in σ.
- Execution of a in σ *causes* f if
  (i) execution of a in σ immediately causes f; or
  (ii) there is a $b \subset a$, such that execution of b in σ immediately causes f, and there is no c such that $b \subset c \subseteq a$ and execution of c in σ immediately causes $\neg f$.
- $E^+(a, \sigma) = \{f : \text{execution of } a \text{ in } \sigma \text{ causes } f\}$
- $E^-(a, \sigma) = \{f : \text{execution of } a \text{ in } \sigma \text{ causes } \neg f\}$
- Definition of transition function, model, and entailment remains the same.

Motivation and Syntax of $\mathcal{AR}$ (adding classical constraints)

- Classical constraints: **always** ϕ

  The formula ϕ should hold in all states.

- **always** $dead \Leftrightarrow \neg alive$

- Recall:

  - $load$ **causes** $loaded$
  - $shoot$ **causes** $\neg alive$ **if** $loaded$
  - **executable** $load$ **if** $\neg loaded$
  - **initially** $\neg loaded$
  - **initially** $alive$

- We would like to conclude $dead$ **after** $load, shoot$

Semantics of $\mathcal{AR}$

- We may now have non-deterministic transitions.
- Two choices
  - Multiple transition function, each encoding a particular non-deterministic choice
    - $toss$ **causes** $tossed$
    - $tossed \Leftrightarrow head \oplus tail$
    - The hypothesis $head$ **after** $toss$ will weed out the model where $tail$ is the result of toss.
    - Unintuitively, we will conclude that $head$ **after** $toss, \alpha$, for any α.
  - A generalization of the notion of transition function
- $\Phi(a, \sigma)$ is a set of states which may be reached by executing a in σ

Characterizing Φ

- Valid state: a state that satisfies all the classical constraints.
- Closeness: Which one between σ_1 and σ_2 is closer to σ.
  $\sigma_1 \leq_\sigma \sigma_2$ iff $((\sigma_1 \setminus \sigma) \cup (\sigma \setminus \sigma_1)) \subseteq ((\sigma_2 \setminus \sigma) \cup (\sigma \setminus \sigma_2))$
- $\Phi(a, \sigma) = \{\sigma' : E^+(a, \sigma) \subseteq \sigma', E^-(a, \sigma) \cap \sigma' = \emptyset$, σ' is valid and there does not exist a valid σ'', which satisfies $\sigma'' <_\sigma \sigma'$ and $E^+(a, \sigma) \subseteq \sigma'', E^-(a, \sigma) \cap \sigma'' = \emptyset$
- Consider the action *shoot* and the state $\sigma = \{alive, loaded\}$
  - $E^+(shoot, \sigma) = \emptyset$; $E^-(shoot, \sigma) = \{alive\}$
  - $(\sigma \cup E^+) \setminus E^- = \{loaded\}$. $\{loaded\}$ is not a valid state.
  - Let $\sigma' = \{dead, loaded\}$. σ' is a valid state.
  - $E^+ \subseteq \sigma'$ and $E^- \cap \sigma' = \emptyset$
  - $\Phi(shoot, \sigma) = \{\sigma'\}$

Inadequacy of classical constr: Qual. or Ramification?

- A block can not be at two different positions at the same time.
  - $at(X) \wedge at(Y) \Rightarrow X = Y$
  - **initially** $at(A)$; $moveto(B)$ **causes** $at(B)$
  - $\neg at(A)$ **after** $moveto(B)$ YES
  - Constraint causes a ramification.
- A person can not be married to two different persons at the same time.
  - $married\_to(X) \wedge married\_to(Y) \Rightarrow X = Y$
  - **initially** $married\_to(A)$; $marry(B)$ **causes** $married\_to(B)$
  - $\neg married\_to(A)$ **after** $marry(B)$ NO.
  - $marry(B)$ is not executable in the initial situation.
  - Constraint blocks qualification.
- Representing constraints in classical logic is not sufficient.

Syntax of $\mathcal{AC}$

- Static causal laws: φ **if** ψ

  $\neg at(X)$ **if** $at(Y), X \neq Y$
- φ **if** $True$ is intuitively equivalent to **always** φ
- $False$ **if** ψ is intuitively equivalent to an integrity constraint $\leftarrow \psi$ (Adds to qualification)

  $False$ **if** $married\_to(X), married\_to(Y), X \neq Y$
- Dynamic causal laws: a **causes** φ **if** ψ

Semantics of $\mathcal{AC}$

- An interpretation (S) of fluents is a maximal consistent set of fluent literals.
- V: a set of fluent formulas.
- R: a set of static causal laws of the form φ **if** ψ.
- V is said to be *logically closed* if it is closed under propositional logic.
- V is said to be closed under R if for every rule φ **if** ψ in R, if ψ belongs to V then so does φ.
- (Note a formula φ is equivalent to the rule φ **if** $true$.)
- $Cn(V \cup R)$ is the least logically closed set of fluent formulas that contains V and is also closed under R.
- An interpretation S is said to be a *state* of D if $Cn(S)$ is closed under the set of static causal laws of D.

- $e_a(S) = \{\varphi \mid D$ contains a law a **causes** φ **if** ψ and ψ holds in $S\}$.
- $\Phi(a, S) = \{S' \mid Cn(S') = Cn((S \cap S') \cup e_a(S) \cup R)$ and S' is a state of $D.\}$

Functional STRIPS

- Difference with STRIPS
  - function symbols are allowed
  - function symbols can be fluents
- Motivation: The Tower of Hanoi problem
  - A STRIPS formulation
    - Relations: $smaller(i, j)$, $on(i, j)$, $clear(i)$
    - actions: $move(i, j, k)$ (i, j, k range over disc names)
    - If # disks (N) = 10, then 1000 ground actions.
    - But interestingly, number of applicable actions in any state are determind by number of pegs, not discs.
      If P pegs then $P \times (P - 1)$ applicable actions.

- A formulation using conditional effects
  * action: $move(p_i, p_j$ – move top of peg p_i to top of peg p_j.
  * Relations: $top(p_i, d_k)$, $on(d_k, d_l)$
  * $move(p_i, p_j)$ **causes** $top(p_i, d_m), top(p_j, d_k), \neg top(p_i, d_k), on(d_k, d_l)$ **if** $top(p_i, d_k), top(p_j, d_l), on(d_k, d_m)$
  * N^3 ground conditional effects associated with each action.
- A functional STRIPS description
  * Domains:
    Peg: p_1, p_2, p_3
    $Disk$: $1 \ldots, 4$
    $Disk*$: $Disk \cup \{0\}$ (0 is a dummy bottom disk)
  * Functional fluents:
    $top : Peg \mapsto Disk*$
    $below : Disk \mapsto Disk*$
    $size : Disk \mapsto Integer$
  * $move(p_i, p_j)$ **causes** $top(p_i) := below(top(p_i)), top(p_j) := top(p_i), below(top(p_i)) := top(p_j)$

* **executable** $move(p_i, p_j$ **if** $top(p_i) \neq 0, size(top(p_i)) < size(top(p_j))$

Formulating Blocks world in an expressive languages

- Initial state: $on(a,b)$, $on(b,table)$
- Goal: $on(a,table)$, $on(b,a)$
- Effect axiom: $move(X,Z)$ **causes** $on(X,Z)$
- Executability conditions: **executable** $move(X,Z)$ **if** $clear(X)$
- Defined fluents:
  (Can not be caused by an effect axiom or be on the left hand side of a ramification constraint)

  $block\_on\_top(X) \leftarrow on(Y,X)$
  $clear(X) \leftarrow not\ block\_on\_top(X)$

- Qualification Constraint:

  $$\leftarrow on(X,Y), on(X',Y), X \neq X', Y \neq table$$

  (Two different blocks X and X' can not be on the same Y, unless Y is the table. If any action violates this condition then that action is not executable.)

- Ramification Constraint:

  $$\neg on(X,Y) \leftarrow on(X,Z), Z \neq Y$$

  (The same block can not be in two different locations Y and Z. If an action violate this it will have an indirect effect.)

Assignment 1. Due date

SEARCH AND ITS ROLE IN PLANNING

Earlier methods

- Forward search (in the state space) from initial state to goal states.
- Backward search (in the state space) from goals to initial state.
- Partial order planning: search in the space of plans.

Heuristic search with domain independent heuristics