Planning With Preferences

Shirin Sohrabi Araghi

(www.cs.toronto.edu/~shirin)

Aug 24, 2005

Supervisor Sheila McIlraith

Want to learn more :

www.cs.toronto.edu/~sheila/pplan/

Acknowledgements – builds on work by students:

Meghyn Bienvenu & Christian Fritz

Motivation

In many real-world settings, **plans are plentiful**.
Challenge: generate **high quality plans** that meet user’s preferences.

The Dinner Example: *It’s dinner time and Claire is tired and hungry. Her goal is to be at home with her hunger sated. There are 3 possible ways for Claire to get food: cook at home, order take-out, or go to a restaurant.*

A straightforward planning problem with many solutions … but ….

- Claire prefers to eat pizza over spaghetti, and spaghetti over crêpes.
- She prefers take-out, followed by cooking at home if she has the ingredients for something she knows how to make. Following this she prefers to go to a restaurant, or finally to cook at home if it requires shopping for groceries.
- If she goes out, she’d like to buy toothpaste, after going to the bank.
- …

Our Challenge

1. How do we **specify** rich user preferences that are relevant to planning?
2. How do we **generate** preferred plans?

1. First Order Preference Language

1. **BDFs (Basic Desire Formulaes)** describe **properties of situations**. They are the **primitive building blocks** of our preferences, and have a Situation Calculus semantics :

- fluent and non-fluent formulae
- final(f), where f is a fluent and occ(a), where a is an acion
- constructions using linear temporal logic (LTL) constructs next(φ), always(φ), eventually(φ), until(φ, ϕ), for BDFs φ, ϕ.
- constructions using variables and FOL constructs $\wedge$, $\vee$, $\neg$, $\forall$, $\exists$

Examples: ($\exists$x).hasIngredients(x) $\wedge$ knowsHowToMake(x) (P1)
($\exists$x).eventually(occ(cook(x))) (P2)
($\exists$x). ($\exists$y).eventually(occ(orderTakeout(x,y))) (P3)

2. **APFs (Atomic Preference Formulaes)** express preferences over BDF properties. Semantics of APFs are defined using weights. An APF is of the form:

$\varphi_0 >> \varphi_1 >> \varphi_2 >> \ldots >> \varphi_n$, where $\varphi_0,\ldots,\varphi_n$ are BDFs

Examples: occ'(eat(pizza)) >> occ'(eat(spag)) >> occ'(eat(crêpes)) (P4)

P1 $\wedge$ P2 >> P3 >> $\neg$ P1 $\wedge$ P2 (P5)

occ'(a) is short for eventually(occ(a))

3. **GPFs (General Preference Formulaes)** provide **syntax for composing preferences** to express preferred combinations of preferences. Semantics of GPFs are defined using weights. GPFs can be:

- ❖ APFs
- ❖ Conditionals: $\gamma : \Psi$, where γ is a BDF and Ψ is a GPF.
- ❖ Negation: $! \, \Psi$
- ❖ And: $\Psi_0 \,\&\, \Psi_1 \,\&\, \ldots \,\&\, \Psi_n$ Or: $\Psi_0 \mid \Psi_1 \mid \ldots \mid \Psi_n$
- ❖ Lex Order: $\Psi_0 \blacktriangleright \Psi_1 \blacktriangleright \ldots \blacktriangleright \Psi_n$

Examples:	P1 : P2	(P6)
	P4 | P5	(P7)
	! P4	(P8)
	P4 ▸ P5	(P9)

(P6) states that if Claire initially has the ingredients for some thing she can make, then she should cook it.

(P7) states that she would be content if either of the two were satisfied.

(P8) states that Claire's most preferred option is eating something other than pizza, spaghetti or crêpes, in that order.

(P9) states that while Claire cares about both her preferences, her food preferences takes priority.

2. Planning with Preferences

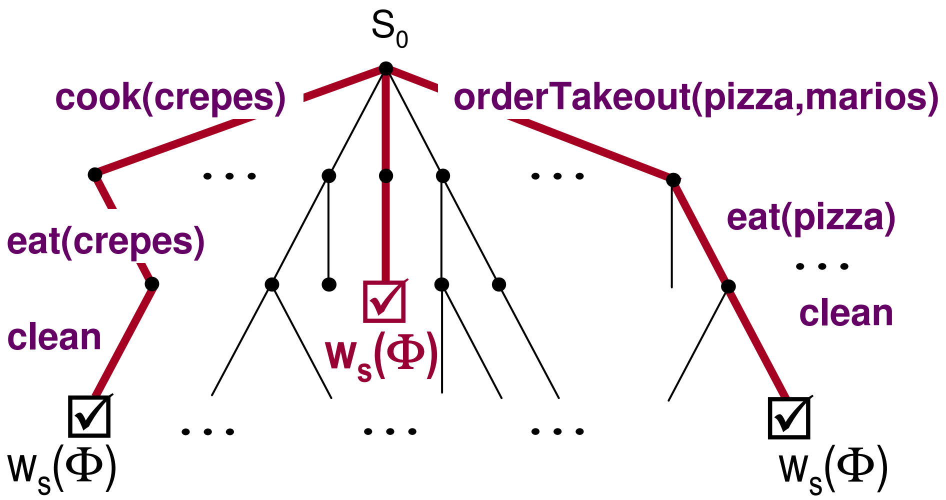
Naïve Approach:

1. Find all sequences of actions that achieve the goal.
2. Compute the weight of each plan trajectory.
3. Choose the plan with the lowest weight.

Inefficient !

Our Approach:

- Bounded,
- Best-First A* Search
- Forward-Chaining Planner
- Implemented in Prolog



Algorithm:

- Compute the optimistic weight of partial plans.
- Progress the preference formula to determine remaining preferences to be achieved.
- Guide search by pursuing the lowest weight partial plan, that has the shortest length.

Accomplishments and Future Work

1. First-Order Preference Language
   - ❖ Expressive Syntax (extending [Son & Pontelli, 2004])
     - ✓ Previous languages propositional and less expressive
   - ❖ Situation Calculus Semantics
     - ✓ Model-theoretic semantic, preferences expressible within the language
     - ✓ Leads to applications beyond planning
2. Forward-Chaining Best-First Search Approach to Computing Preferred Plans
   - ✓ Proved correct and optimal
   - ✓ Doesn't require computation and comparison of all possible plans
   - ✓ Exploits progression (Theorems confirm preservation of semantics)
   - ✓ Experiments illustrate effectiveness of best-first search heuristic
   - ✓ Code and domains available for experimentation and reuse

Future Work : I will be taking a fourth year course (CSC494) in which I will:

- ➢ Extend the planner to agent programming
- ➢ Apply my work to automated Web service composition
- ➢ Rethink the semantics of our preference language.