Planning with Temporal Logic

April 25, 2016
Motivation

• Consider a self-driving car…

• Regardless of our destination, we also want to make sure we always follow the rules of the road.
Key Takeaways

• Modeling temporally-extended goals with linear temporal logic (LTL)

• Modeling preferences between alternative plans
Outline

• Introduction to Linear Temporal Logic
  – Why use Linear Temporal Logic?
  – Linear Temporal Logic Operators
  – Example LTL Problems

• Applications to Planning

• Planning with Preferences
  – Expressing Preferences
  – Planning in LPP
Linear Temporal Logic
Temporal Logic

• Formalism for specifying properties of systems that vary with time
Temporal Logic

• Systems proceed through a sequence of discrete states

![Diagram showing a sequence of states A, B, C, D with arrows indicating transitions between them.]

...
Why Temporal Logic?

• Previously our planning algorithms have used propositional logic to specify goals dealing with a single state at a single point in time.

• Temporal logic allows these goals to be specified over a sequence of states.
Why Temporal Logic?

• What if the problem requires a condition to:
  – Be met until another condition is met...
    • For example: red implies (stop until green)
Why Temporal Logic?

• What if the problem requires a condition to:
  – Always eventually be met
  • For example, always have some point in the future when you visit a gas station
Temporal Logic

Branching vs linear time

- Linear time
  - Models physical time
  - At each time instant, only one of the future behaviors is considered
  - We can reason about always
Temporal Logic

Branching vs linear time

• Branching time
  – At each time instant, all possible future behaviors are considered
  – Time may split into alternate courses
  – We can reason about possibilities
Branching vs linear time

• Linear time

• Branching time
Linear Temporal Logic (LTL) involves:

- Linear time model
- Infinite sequences of states
- Forward-looking conditions

Cannot express properties over a set of different paths
Applications of Temporal Logic

• Temporal logic is used in:
  – Verification and Model Checking
    • Safety and Maintenance
  – Planning
LTL Syntax

LTL formula \( f := \text{true} \mid p_i \mid f_i \land f_j \mid \neg f_i \mid X f_i \mid f_i U f_j \)

An LTL formula is built from:

1. Propositional variables: \( p, \rho, \phi, \omega \) etc.
   - Can be True or False
2. Logical Operators: \( \neg, \lor, \land, \rightarrow, \leftrightarrow, \text{True, False} \)
   
   - \( \neg = \text{not} \)
   - \( \lor = \text{or} \)
   - \( \land = \text{and} \)
   - \( \rightarrow = \text{implies} \)
   - \( \leftrightarrow = \text{if and only if} \)
   - \( \text{True, False} \)
Logical Operator Examples

<table>
<thead>
<tr>
<th>Logical Operators</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>$p = \text{true}$</td>
<td>$R = \text{red light}$</td>
</tr>
</tbody>
</table>
Logical Operator Examples

**Logical Operators**

**Example**

not, $\neg$

$\neg G = \text{green light}$

---

**Logical Operators**

**Example**

and, $\land$

$R \land B = \text{gas station}$
### Logical Operator Examples

<table>
<thead>
<tr>
<th>Logical Operators</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>or, $\lor$</td>
<td>$R \lor G$</td>
</tr>
</tbody>
</table>

Or ($\lor$) can be rewritten with and ($\land$) and not ($\neg$)

$$R \lor G = \neg(\neg R \land \neg G)$$

Similar process can be done for implies and iff, but we won’t be explaining them due to time constraints.
LTL Syntax

LTL formula $f := \text{true} \mid p_i \mid f_i \land f_j \mid \neg f_i \mid X f_i \mid f_i U f_j$

An LTL formula is built from:

1. Propositional variables: $p, \rho, \phi, \omega$ etc.
   Can be True or False
2. Logical Operators: $\neg, \lor, \land, \rightarrow, \leftrightarrow, \text{True}, \text{False}$
   
   $\neg = \text{not}$
   $\lor = \text{or}$
   $\land = \text{and}$
   $\rightarrow = \text{implies}$
   $\leftrightarrow = \text{if and only if}$
   $\neg\text{True, False}$

3. Temporal Operators
Temporal Operators

What are some useful operators we may want to describe our car?
Temporal Operators

• The **next** light to be **green**

• The light will be **red until it is green**

• The light will **eventually** turn **green** some point in the **future**, turn **green**
Temporal Operators

• The light will **always** be **red**

• The light will be **red** until the car gets **gas** and the state after it’s **released**, the light can be whatever
<table>
<thead>
<tr>
<th>Operator</th>
<th>Textual Operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>ne\textit{X}t</td>
<td>$X_\rho$</td>
</tr>
</tbody>
</table>

*Definition*: Variable $\rho$ must be true in the next state
**Definition:** Variable $\rho$ must remain true up until the state where variable $\omega$ becomes true, at which point $\rho$ becomes unconstrained.

Note that $\omega$ is required to become true in some future state.
Future

<table>
<thead>
<tr>
<th>Operator</th>
<th>Textual Operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Future/Eventually</td>
<td>$F\rho$</td>
</tr>
</tbody>
</table>

*Definition*: Variable $\rho$ must become true in some future state

![Diagram](image-url)
Global

<table>
<thead>
<tr>
<th>Operator</th>
<th>Textual Operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Globally</td>
<td>$G\rho$</td>
</tr>
</tbody>
</table>

*Definition*: Variable $\rho$ must be true in all future states
**Release**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Textual Operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release</td>
<td>$\rho R \omega$</td>
</tr>
</tbody>
</table>

**Definition:** Variable $\rho$ must be true up until and including the state where $\omega$ becomes true, after which $\omega$ is unconstrained. If $\rho$ is not true in any future state, then $\omega$ is true in all future states.

Different from $U$ in that both $\rho$ and $\omega$ are true in one state.
Which describe the other?

**Future/Eventually**

**Release**

**Globally**

\[
\equiv \text{True U } \rho \\
\equiv \neg F \neg \rho \\
\equiv \neg (\neg \rho \ U \ \neg \omega)
\]
Which describe the other?

**Future/Eventually**

**Release**

**Globally**

\[
\equiv \text{True U } \rho \\
\equiv \neg \text{F}\neg \rho \\
\equiv \neg (\neg \rho \text{ U } \neg \omega)
\]
<table>
<thead>
<tr>
<th>Operator</th>
<th>Textual Operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textsc{next}</td>
<td>\textit{X}_\rho</td>
</tr>
<tr>
<td>\textsc{Until}</td>
<td>\rho U \omega</td>
</tr>
<tr>
<td>\textsc{Future/Eventually}</td>
<td>\textit{F}_\rho \equiv \text{True U } \rho</td>
</tr>
<tr>
<td>\textsc{Globally}</td>
<td>\textit{G}_\rho \equiv \neg \text{F}\neg\rho</td>
</tr>
<tr>
<td>\textsc{Release}</td>
<td>\rho R \omega \equiv \neg(\neg \rho U \neg \omega)</td>
</tr>
</tbody>
</table>
Combination of Operators

Infinitely Often

\[ GFp \rightarrow \cdots \rightarrow p \rightarrow \cdots \]

Eventually Forever

\[ FGp \rightarrow \cdots \rightarrow p \rightarrow p \rightarrow \cdots \]
Example Problem

What are some true statements about this LTL formation?

- $XR$
- $FG$
- $RUG$
- $(RUG) \land (FG) \land (XR)$
Expressing Temporal Logic in PDDL

PDDL3 Goal Description

<GD> ::= (at end <GD>)
| (always <GD>)
| (sometime <GD>)
| (within <num> <GD>)
| (at-most-once <GD>)
| (sometime-after <GD> <GD>)
| (sometime-before <GD> <GD>)
| (always-within <num> <GD> <GD>)
| (hold-during <num> <num> <GD>) | ...
## Temporal Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>PDDL3</th>
</tr>
</thead>
<tbody>
<tr>
<td>neXt</td>
<td>$\text{X}_\rho$ (within 1 $\rho$)</td>
</tr>
<tr>
<td>Until</td>
<td>$\rho U \omega$ (always-until $\rho \omega$)</td>
</tr>
<tr>
<td>Future</td>
<td>$\rho F \omega$ (sometime-after $\rho \omega$)</td>
</tr>
<tr>
<td>Globally</td>
<td>$G\rho$ (always $\rho$)</td>
</tr>
<tr>
<td>Release</td>
<td>$\rho R \omega$ (or)</td>
</tr>
<tr>
<td></td>
<td>(always $\omega$)</td>
</tr>
<tr>
<td></td>
<td>(always-until $\omega \rho$))</td>
</tr>
</tbody>
</table>
• The traffic light will turn red in the next state

\[
\begin{align*}
\text{(goal (within 1 (turn red)))}
\end{align*}
\]

Command Syntax

\[
\begin{align*}
(\text{within } \langle \text{num} \rangle \ \langle \text{GD} \rangle)
\end{align*}
\]

\[
\begin{align*}
(\text{within } \langle \text{num} \rangle \ \varphi) \ would \ mean \ that \ \varphi \ must \ hold \ within \\
\langle \text{num} \rangle \ \text{happenings}
\end{align*}
\]
The traffic light will be green until it turns red at which point it will be red forever:

\[(g \mathcal{U} r) \land (r \rightarrow G r)\]

(:goal
  (and
   (always-until (turn green) (turn red))
   (implies (turn red) (always (turn red))))
))
Application to Planning

Problem with Temporarily Extended Goals

LTL or PDDL3

Buchi Automata

PDDL2

Plan

Classical Planner

Planner with Progression
A Büchi automaton is a 5-tuple $<S, s_0, T, F, \Sigma>$

- $S$ is a finite set of states
- $s_0 \in S$ is an initial state
- $T \subseteq S \times \Sigma \rightarrow S$ is a transition relation
- $F \subseteq S$ is a set of accepting states
- $\Sigma$ is a finite set of symbols ('alphabet')

An infinite sequence of states is accepted iff it visits the accepting state(s) infinitely often.
Example Büchi Automata

Example: Model a clock

Accepted words:

Tick Tock Tock Tick Tock Tick Tick Tick Tick Tock...

Tock Tock Tick Tock Tick Tock Tock Tock Tock Tock...
Example Büchi Automata

Example: Model a clock

Accepted words:

TockTickTickTickTickTickTickTickTickTick...
Example Büchi Automata

Example: Model a clock

Accepted words:
TockTickTockTickTockTickTockTickTickTick...
LTL to Büchi Automata

next?
Future/Eventually?
Globally?
**Future** - $Fp \equiv \text{True} \cup p$

Accepted word: $\neg p \ 
eg p \ 
eg p \ p \ p \ 
eg p \ ...$

Sequence of states: $s_0 \ s_0 \ s_0 \ s_1 \ s_1 \ s_1 ...$

**Globally** - $Gp \equiv \neg F \neg p$

Accepted word: $p \ p \ p \ p \ p \ p ...$

Sequence of states: $s_0 \ s_0 \ s_0 \ s_0 \ s_0 \ s_0 ...$
LTL to Büchi Algorithm

\[
\begin{align*}
N &\quad \text{Node object} \\
N.\text{curr} &\quad \text{LTL formulas to be processed} \\
N.\text{old} &\quad \text{LTL formulas already processed} \\
N.\text{next} &\quad \text{LTL formulas to be processed in next node} \\
N.\text{incoming} &\quad \text{Incoming transitions from predecessor nodes} \\
N_\text{s} &\quad \text{List of processed Nodes} \\
N_\text{i} &\quad \text{Arbitrary node from } N_\text{s} \\
\end{align*}
\]

\textbf{expand} \ (N,N_\text{s})

\begin{align*}
\text{if } N.\text{curr} &\text{ is empty} \\
\text{if } N.\text{curr} = N_\text{i}.\text{curr} &\text{ Append } N.\text{curr} \text{ to } N_\text{i}.\text{curr} \\
\text{else} &\text{ Append } N \text{ to } N_\text{s} \\
&\text{Create new node } N_\text{new} \text{ with } N_\text{new}.\text{curr} = N.\text{next} \\
&\text{expand} \ (N_\text{new},N_\text{s}) \\
\text{else} &\text{ Remove an LTL formula } f \text{ from } N.\text{curr} \text{ and append to } N.\text{old} \\
&\text{Perform Progression on } f \\
&\text{Call expand on result of Progression}
\end{align*}

The result of this algorithm is a generalized Büchi automata which is then transformed into a simple Büchi automata.
Progression Algorithm

\textbf{progress}(f, N, Δt = 1) \#Δt is time between successive states

if \( f \) contains no temporal qualities:
  if \( N\text{.curr entails } f \):
    \( f' = \text{True} \)
  else
    \( f' = \text{False} \)

if \( f = f_1 \land f_2 \):  
  \textbf{progress}(f_1, N, Δt) \land \textbf{progress}(f_2, N, Δt)

if \( f = Xf_1 \):
  \( N\text{.next.append}(f_1) \)

if \( f = f_1 U_{[a,b]} f_2 \):
  \#[a,b] is a time interval that could be infinite
  if \( b < a \):
    \( f' = \text{False} \)
  else if \( 0 \in [a,b] \):
    \textbf{progress}(f_2, N, Δt) \lor \left( \textbf{progress}(f_1, N, Δt) \land N\text{.next.append}(f_1 U_{[a,b]} - Δt f) \right)
  else
    \textbf{progress}(f_1, N, Δt) \land N\text{.next.append}(f_1 U_{[a,b]} - Δt f)
Büchi states are not equivalent to PDDL2 states. Consider:

$$\text{FutureGlobally} \cdot F \text{Gp}$$

Two ways to transform temporally extended goals to PDDL2:

- Create new actions that encapsulate the allowable transitions in each state
- Introduce derived predicates
  - Do not depend on the actions
  - Used to determine which state the planner is in
  - Goal of the planner is to move from initial state to any accepting state
Planning with Preferences
Preference Based Planning

Classical Planning Problem

\[ \text{problem} := (S, s_0, A, G) \]

- **\( S \)** - set of states
- **\( s_0 \)** - initial state
- **\( A \)** - set of operators
- **\( G \)** - set of goal states

Preference-based Planning Problem

\[ \text{problem} := (S, s_0, A, G, R) \]

- **\( R \)** is a partial or total relation expressing preferences \((\preceq)\) between plans

Preferences express properties of the plan that are desired but not required
Preference Expression Languages

- **Quantitative** - assign numeric values to plans to compare them
  - Markov Decision Processes (MDP’s)
    - Find preferred policy using a reward function over conditional plans
  - PDDL3
    - Preferences expressed through reward function based on satisfying/violating logical formulas on the plan

- **Qualitative** - relations compare plans based on properties of the plans that need not be numeric
  - Ranked Knowledge Bases
    - Plan properties are ranked with preferred formulas ranked higher
  - Temporally Extended Preferences
    - Use LTL to express plan properties that are then ranked

Quantitative languages imply total comparibility while qualitative languages may allow incomparability
Expressing Preferences in PDDL3

Syntax for modeling preferences:
(preference [name] <GD>) - label for fluents that represent preferences

is-violated - function that returns the number of times the preference was not satisfied in the plan

Example:

\textit{Traffic light is green until it turns red}

(preference gUr
  (always-until (turn green) (turn red)))

\textit{Plan tries to not violate any preferred fluents}

(metric minimize (is-violated gUr))
LPP Language Overview

• LPP is a quantitative language to express temporal preferences for planning
  – Preferences between different temporal goals can be expressed along with the strength of preference
    • i.e. Goal A is preferred twice as much as Goal B
• LPP is an extension of an older language PP
• Preference formulas in LPP are constructed hierarchically

Constructing a Preference Formula

**Basic Desire Formula (BDFs)** express temporally extended propositions

- At some point, will cook  
  \[- b_1 = F(\text{cook})\]
- At some point, will order takeout  
  \[- b_2 = F(\text{orderTakeout})\]
- At some point, will eat spaghetti  
  \[- b_3 = F(\text{eatSpaghetti})\]
- At some point, will eat pizza  
  \[- b_4 = F(\text{eatPizza})\]
Atomic Preference Formulas (APFs) express preferences between BDFs

• In this example, weights associated with each BDF define preferences
  – Lower weight is preferred

• Prefer to cook over ordering takeout
  – \( a_1 = b_1[0.2] \gg b_2[0.4] \)

• Prefer eating spaghetti over eating pizza
  – \( a_2 = b_3[0.3] \gg b_4[0.9] \)
**General Preference Formulas (GPFs)**

allow conjunctions or disjunctions of APFs or qualification of BDFs with conditionals

- Satisfy the most preferred option among the APFs (satisfy APF with lowest weight)
  
  \[ g_1 = a_1 | a_2 \]

- Choose the most preferred option that satisfies both APFs (minimize the maximum weight across both APFs)
  
  \[ g_2 = a_1 & a_2 \]
Aggregated Preferences Formulas (APFs) define the order in which preferences should be relaxed

• Prefer that if both $g_1$ and $g_2$ from previous slide can’t be met, that $g_2$ from previous slide is met
  $$\neg g_1 \land g_2 \preceq g_2 \preceq g_1$$

• Situations that aren’t distinguished any other way can be sorted lexicographically (alphabetically)
LPP Formula Hierarchy Review

• Basic Desire Formula (BDF)
  — Express temporally extended propositions

• Atomic Preference Formula (APF)
  — Express preferences between BDFs

• General Preference Formula (GPF)
  — Allow conjunctions or disjunctions of APFs or qualification of BDFs with conditionals

• Aggregated Preference Formula (APF)
  — Define the order in which preferences should be relaxed


Appendix
Solving Planning Problems with Preferences

• **PPLAN**
  – implemented by Meghyn Bienvenu, Christian Fritz, and Sheila A. McIlraith

• Solves planning problems with preferences expressed in LPP via bounded best-first search forward chaining planner
  – use of *progression* efficiently evaluates how well partial plans satisfy Φ (a general preference formula)
  – use of admissible *evaluation function* ensures best-first search is optimal
Quick Definitions

• Forward Chaining Planner - Forward chaining starts with the available data and uses inference rules to extract more data (from an end user, for example) until a goal is reached.

• A situation $s$ is a history of the primitive actions $a \in A$ performed from an initial situation $S_0$. 
Progression

• Purpose of progression:
  – take in a situation and temporal logic formula (TLF)
  – evaluates the TLF with respect to the state of the situation
  – generates a new formula representing those aspects of the TLF that remain to be satisfied in subsequent situations.

• Weight of general preference formula with respect to a situation is equal to progressed preference formula with respect to final situation
Evaluation Function

- Evaluation Function
  - has optimistic and pessimistic weights to provide best and worst weights on a successor with respect to $\Phi$.
  - the optimistic weight is non-decreases and does not over-estimate the actual weight
  - this allows PPLAN to define an optimal search algorithm
PPLAN Algorithm

optW = optimistic weight (Assumes all unfulfilled preferences are fulfilled)
pessW = pessimistic weight (Assumes all unfulfilled preferences are not fulfilled)

Algorithm
L = list of nodes sorted by optW, then pessW, then length

while L is not empty
    Remove first node from L
    If goal is achieved and optW = pessW
        return partial plan, optW
    Perform Progression
    Add new nodes to L and sort
PPLAN

• PPLAN is implemented with a
  – general preference formula $f\Phi$ they define is
    admissible and when used in best first
    search, the search is optimal
  – the best first search searches through the
    partial plans based on their weights
  – for full details see paper "Planning with
    Qualitative Temporal Preferences" by Fritz,
    Christian, Sheila A. McIlraith, and Meghyn
    Bienvenu.
Additional Examples were taken from the youtube videos of NOC15 July-Oct CS12:
https://www.youtube.com/watch?v=W5Q0DL9plns
Example Formulations

• Traffic light is red: \( r \)
• Traffic light is green: \( g \)

• The traffic light will turn red in the next state
  \( \neg Xr \)
• The traffic light will be green until it turns red but it may not ever turn red
  \( (g \ U \ r) \lor Gg \) (Weak Until)
• The traffic light will be green until it turns red at which point it will be red forever
  \( (g \ U \ r) \land (r \rightarrow Gr) \)
Additional Examples

\[
\begin{align*}
\phi &:= \text{true} | p_i | \phi_1 \land \phi_2 | \neg \phi_1 | X \phi | \phi_1 U \phi_2 \\
p_i &\in AP \\
\phi_1, \phi_2 &\colon \text{LTL formulas}
\end{align*}
\]
\[
\phi := \text{true} \mid p_i \mid \phi_1 \land \phi_2 \mid \neg \phi_1 \mid X \phi \mid \phi_1 U \phi_2
\]

\[\neg (p_1 U p_2)\]
\[\{p_1\} \quad \{p_1\} \quad \{\} \quad \{p_2\} \quad \{p_1\} \quad \ldots\]

\[p_1 U (p_2 \land X p_3)\]
\[\{p_1, p_3\} \quad \{p_1\} \quad \{p_1\} \quad \{p_2\} \quad \{p_1, p_3\} \quad \ldots\]

\[X (\neg p_1 U p_2)\]
\[\{p_1\} \quad \{\} \quad \{\} \quad \{p_2\} \quad \{p_1\} \quad \ldots\]

\[\text{true} U p_1\]
\[\{p_2\} \quad \{p_3\} \quad \{p_2\} \quad \{\} \quad \{p_1\} \quad \ldots\]

\[\neg (\text{true} U \neg p_1)\]
\[\{p_1\} \quad \{p_1, p_2\} \quad \{p_1\} \quad \{p_1, p_2\} \quad \{p_1\} \quad \ldots\]