Concurrent Plan Recognition &
Execution for Human-Robot Teams

Cognitive Robotics 2016 Lecture

Steven J. Levine

Wednesday, March 16th, 2016
Intent recognition & adaptation are siblings

- Intent recognition & robot adaptation are both necessary to build intelligent robots that work with people

© source unknown. All rights reserved. This content is excluded from our Creative Commons license. For more information, see [https://ocw.mit.edu/help/faq-fair-use/](https://ocw.mit.edu/help/faq-fair-use/).
Intent recognition & adaptation are siblings

• Intent recognition & robot adaptation are both necessary to build intelligent robots that work with people
Intent recognition & adaptation must be integrated

**Intent Recognition**

- Kautz and Allen 1986
- Avrahami-Zilberbrand, Kaminka, and Zarosim 2005
- Wu, Osuntogun, Choudhury, Philipose, Rehg 2007
- Freedman, Jung, and Zilberstein 2014
- Goldman, Geib, and Miller 1999
- Ramirez & Heffner 2009
- Smith, Shah, da Vitoria Lobo 2004
- Song, Demirdjian, Davis 2011

*Gesture, Sketch, & Pose Recognition*

... and many more!

**Robot Adaptation**

- TPOPExec (Muise, Beck, and McIlraith 2013)
- HATP + SHARY
- Dechter, Meiri, Pearl 1991
  - Chaski (Shah, Conrad, Williams 2009)
- Drake
  - (Conrad, Shah, and Williams 2009)
- Finzi, Ingrand, and Muscettola 2004
  - Bui 2003
-蛤类, Junk, and Zilberstein 1998
- Morris 1998
  - Effinger et. al 2009
  - Hofmann et. al. 2005
- HOTRIDE
  - Ayan et al. 2007
- IPEM
  - Ambros-Ingerson and Steel 1998
- Vidal 1999
  - Tedrake, Manchester, Tobenkin, Roberts 2010
- T-Rex
  - Py, Rajan, and McGann 2010
  - IxTeT eXeC
    - (Lemai and Ingrand 2004)
- ROGUE
  - (Haigh and Veloso 1998)

*Control theory*

... and many more!

**Our contribution:**

*Pike*

Concurrent Plan Recognition & Execution for Human-Robot Teams  Steven J. Levine
Intent recognition & adaptation must be integrated

• Much prior work on intent recognition, and on robotic adaptation, but largely as separate research

• We present a **unified approach** to plan recognition & robotic adaptation for plans with choice
  • Single algorithm concurrently achieves both
  • **Result**: mixed-initiative execution where robots & humans work together as team
Pike: an executive for human-robot teams

• Given a plan with choice (contingent, temporally flexible):
  • Make decisions online (consistent with human’s intent)
  • Dispatch activities at proper times
  • Monitor execution for problems
How to recognize intent & adapt?

Intent recognition is recognizing decisions consistent with team’s task goals.

Robot adaptation is making decisions consistent with team’s task goals.

- Assume rational, cooperative agents

- Prune any (irrational) decisions resulting in plan failure:
  - Unmet action preconditions: \[ \Rightarrow \text{Causal link reasoning} \]
  - Missed deadlines: \[ \Rightarrow \text{Temporal conflicts} \]
  - Unanticipated failures: \[ \Rightarrow \text{Online execution monitoring} \]
Approach in a nutshell

- Key to our approach:
  - Plan representation with choices & actions for human, robot
Contingent, temporally-flexible plans

Temporal Planning Network with Uncertainty (TPNU)
Contingent, temporally-flexible plans
Temporal Planning Network with Uncertainty (TPNU)
Temporal Planning Network with Uncertainty (TPNU)

This candidate subplan is temporally inconsistent.

Conflict: \( \neg(x_{A2} = \text{coffee} \land x_{A4} = \text{bagel}) \)

(Conrad 2009)
Contingent, temporally-flexible plans

Temporal Planning Network with Uncertainty (TPNU)
First part: making a drink

- Get mug
- Get glass
- Get grounds
- Get juice
- Make coffee
- Pour coffee mug
- Pour juice glass
Extracting labeled causal links

- Get mug
- Get glass
- Get grounds
- Get juice
- Make coffee
- Pour coffee mug
- Pour juice glass

(have mug): \( x_{A1} = \text{mug} \)
(have glass): \( x_{A1} = \text{glass} \)
(have grounds): \( x_R = \text{grounds} \)
(have juice): \( x_R = \text{juice} \)
Suppose person picks up mug…

\[
\begin{align*}
\text{Get mug} & : (\text{have mug}) : \{x_{A1} = \text{mug}\} \\
\text{Get grounds} & : (\text{have grounds}) : \{x_R = \text{grounds}\} \\
\text{Get juice} & : (\text{have juice}) : \{x_R = \text{juice}\} \\
\text{Make coffee} & \\
\text{Pour coffee mug} & \\
\text{Pour juice glass} &
\end{align*}
\]
...so can’t pour juice later...
...so robot should get coffee now.
• **Intent Recognition**: recognizing human’s choices consistent with *at least one* team subplan

• **Robot adaptation**: making robot's choices consistent with *at least one* remaining team subplan
Pike in larger architecture

- **Activity Recognizer**: observes human choices
- **State estimator**: reports current world state
- **Activity Dispatcher**: calls lower-level planning & execution

Contingent, temporally-flexible Plan
Offline

Labeled APSP

Extract Labeled Causal Links

Compile Constraints

Online

Execute Plan

\[
\begin{align*}
z &= 1 \Rightarrow (a_c &= 1 \lor (a_c = 2 \land y = 1)) \\
\neg (a_c &= 1 \land x = 2 \land z = 1)
\end{align*}
\]
Concurrent Plan Recognition & Execution for Human-Robot Teams

Offline

Labeled APSP

Extract Labeled Causal Links

Compile Constraints

Online

Execute Plan

Concurrent Plan Recognition & Execution for Human-Robot Teams  I  Steven J. Levine
Causal links justify action preconditions

- Insufficient for contingent, temporally-flexible plans:
  - What if producer doesn’t execute?
  - What if consumer doesn’t execute?
  - Determining ordering is non-trivial

- We generalize to ***labeled causal links***
  - Encode requisite choices for causal link to hold
Labeled causal links

\[
\begin{align*}
x = 1 & \quad \text{make } p \\
x = 2 & \quad \text{require } p
\end{align*}
\]
Labeled causal links

\[
\begin{align*}
x = 1 & \quad \text{make } p \\
x = 2 & \quad \text{require } p
\end{align*}
\]
Causal link extraction in a nutshell*

• For each precondition of each \textit{consumer} event:
Causal link extraction in a nutshell*

- For each precondition of each **consumer** event:
  - Find all **producers** provably before or during consumer
Causal link extraction in a nutshell*

- For each precondition of each **consumer** event:
- Find all **producers** provably before or during consumer

\[ p : \{ a_c = c_1 \} \]
\[ y = 1 \]
\[ y = 2 \]
\[ z = 1 \]
\[ z = 2 \]
\[ p : \{ a_c = c_2 \land y = 1 \} \]
\[ p : \{ a_c = c_3 \} \]
Causal link extraction in a nutshell*

- For each precondition of each **consumer** event:
  - Find all **producers** provably before or during consumer
  - Add propositional & temporal constraints for each producer

\[
p : \{a_c = c_1\}
\]

\[
y = 1 \quad \text{make} \ p
\]

\[
y = 2
\]

\[
z = 1 \quad \text{require} \ p
\]

\[
z = 2
\]

\[
p : \{a_c = c_2 \land y = 1\}
\]

\[
p : \{a_c = c_3\}
\]

\[
[0, \infty]
\]

\[
\text{make} \ p
\]
Causal link extraction in a nutshell*

- For each precondition of each **consumer** event:
  - Find all **producers** provably before or during consumer
  - Add propositional & temporal constraints for each producer
Causal link extraction in a nutshell*

- For each precondition of each consumer event:
  - Find all producers provably before or during consumer
  - Add propositional & temporal constraints for each producer

Constraint:  \((z = 1) \Rightarrow [(a_c = c_1) \vee (a_c = c_2 \land y = 1) \vee (a_c = c_3)]\)
Causal link extraction in a nutshell*

• For each precondition of each consumer event:
  • Find all producers provably before or during consumer
  • Add propositional & temporal constraints for each producer
  • Find all potential threats probably before or during causal link
Causal link extraction in a nutshell*

- For each precondition of each **consumer** event:
  - Find all **producers** provably before or during consumer
  - Add propositional & temporal constraints for each producer
  - Find all potential threats probably before or during causal link
Causal link extraction in a nutshell*

- For each precondition of each **consumer** event:
  - Find all **producers** provably before or during consumer
  - Add propositional & temporal constraints for each producer
  - Find all potential threats probably before or during causal link
Causal link extraction in a nutshell*

- For each precondition of each consumer event:
  - Find all producers provably before or during consumer
  - Add propositional & temporal constraints for each producer
  - Find all potential threats probably before or during causal link

\[
(z = 1) \Rightarrow \left[ (a_c = c_1) \lor (a_c = c_2 \land y = 1) \right]
\]
Causal link extraction in a nutshell*

- For each precondition of each consumer event:
  - Find all producers provably before or during consumer
  - Add propositional & temporal constraints for each producer
  - Find all potential threats probably before or during causal link
  - Resolve via additional propositional & temporal constraints
Causal link extraction in a nutshell*

- For each precondition of each **consumer** event:
  - Find all **producers** provably before or during consumer
  - Add propositional & temporal constraints for each producer
  - Find all potential threats probably before or during causal link
  - Resolve via additional propositional & temporal constraints

\[ p : \{a_c = c_1\} \]

\[ x = 1 \]

\[ x = 2 \]

\[ y = 1 \]

\[ y = 2 \]

\[ z = 1 \]

\[ z = 2 \]

**Constraint:** \( \neg (z = 1 \land a_c = c_1 \land x = 2) \)
Causal link extraction in a nutshell*

- For each precondition of each consumer event:
  - Find all producers provably before or during consumer
  - Add propositional & temporal constraints for each producer
  - Find all potential threats probably before or during causal link
  - Resolve via additional propositional & temporal constraints

\[ p : \{a_c = c_2 \land y = 1\} \]

\[ b = \begin{cases} 
1 & \text{if threat comes before} \\
2 & \text{if threat comes after} 
\end{cases} \]

\[ x = 1 \]

\[ x = 2 \]

\[ \neg p \]

\[ [0, \infty] \]
Causal link extraction in a nutshell*

• For each precondition of each **consumer** event:
  • Find all **producers** provably before or during consumer
  • Add propositional & temporal constraints for each producer
  • Find all potential threats probably before or during causal link
  • Resolve via additional propositional & temporal constraints

\[
p : \{a_c = c_2 \land y = 1\}
\]

\[
[0, \infty] : \{b = 1 \land z = 1 \land a_c = c_2 \land y = 1 \land x = 2\}
\]

\[
x = 1
\]

\[
x = 2
\]
Causal link extraction in a nutshell*

- For each precondition of each **consumer** event:
  - Find all **producers** provably before or during consumer
  - Add propositional & temporal constraints for each producer
  - Find all potential threats probably before or during causal link
  - Resolve via additional propositional & temporal constraints

\[
\begin{align*}
\{a_c = c_2 \land y = 1\} \\
\{b = 2 \land z = 1 \land a_c = c_2 \land y = 1 \land x = 2\}
\end{align*}
\]
Constraints

\[\phi_{ec} \Rightarrow \bigvee_i (a_i = e_{pi} \land \phi_{epi})\]

\[[\epsilon, \infty] : \{a_{ec\cdot p} = e_{pi}\} \land \phi_{epi} \land \phi_{ec}\text{ from } e_{pi}\text{ to } e_c\]

\[\neg \phi_{C}\]

\[[\epsilon, \infty] : \phi_{C}\text{ from } e_c\text{ to } e_{ti}\]

\[[\epsilon, \infty] : \phi_{C}\text{ from } e_{pi}\text{ to } e_{ti}\]

\[[\epsilon, \infty] : \{b_{ec,p,e_{pi},e_{ti}} = 1\} \land \phi_{C}\text{ from } e_{ti}\text{ to } e_{pi}\]

\[[\epsilon, \infty] : \{b_{ec,p,e_{pi},e_{ti}} = 2\} \land \phi_{C}\text{ from } e_c\text{ to } e_{ti}\]

One candidate causal link must hold
Producers precede consumers
Temporal conflicts (Conrad 2009)
Threat resolutions
Threat resolutions
Threat resolutions

\[\downarrow\]

- Constraints satisfied: team success!
- Preconditions of all executed actions met
- No missed deadlines
Offline

Labeled APSP

Extract Labeled Causal Links

Compile Constraints

Online

Execute Plan

\[
\begin{align*}
z = 1 & \Rightarrow (a_c = 1 \lor (a_c = 2 \land y = 1)) \\
\neg(a_c = 1 \land x = 2 \land z = 1)
\end{align*}
\]
Compile constraints with ATMS

• Compile constraints for fast, online reactivity.

• Assumption-based Truth Maintenance System (ATMS): knowledge base permitting fast querying of assumptions

• Fast online queries without re-solving CSP:
  • “Can robot pick up coffee grounds now?”
  • “Is plan still feasible?”

• Internally, employs label propagation to pre-compute sets of consistent solutions
**Offline**

Labeled APSP

Extract Labeled Causal Links

Compile Constraints

**Online**

Execute Plan
Online execution

- Similar architecture to (Conrad 2009)
- Continually iterates over all events:
  - Gathers constraints necessary to execute event now
  - Queries ATMS: can commit to constraints?
    - If yes: execute event & dispatch appropriate activities
- Receives human’s choice outcomes (from activity recognizer)
- Monitors active causal links
  - Upon violation: add appropriate constraints to ATMS
  - Execution infeasible? Signal execution error.
Online causal link execution monitoring

• Detects potential problems immediately

• Allows recovery actions (if modeled in plan)
Online causal link execution monitoring

- Detects potential problems immediately
- Allows recovery actions (if modeled in plan)
Online causal link execution monitoring

- Detects potential problems immediately
- Allows recovery actions (if modeled in plan)

Disturbance: ¬p

Slide 47
Experimental results

- Randomly-generated TPNU’s with randomly-generated causal link structure (probably harder)
- Compilation time roughly proportional to candidate subplans, (large variance)
- Reactive online performance
Offline

Labeled APSP

Extract Labeled Causal Links

 Compile Constraints

Online

Execute Plan

\[\begin{align*}
z &= 1 \Rightarrow (a_e = 1 \lor (a_e = 2 \land y = 1)) \\
\neg (a_e = 1 \land x = 2 \land z = 1)
\end{align*}\]
Labeled all-pairs shortest path (APSP)

- Developed in (Conrad 2009)

- Three purposes:
  1. Dispatchable form for online execution
  2. Ordering over events for causal link extraction
  3. Temporal conflict extraction
Labeled all-pairs shortest path (APSP)

• What is temporal distance between these events?
Labeled all-pairs shortest path (APSP)

- What is temporal distance between these events?
- Event dependent: \[ \begin{cases} [6, 11] & \text{if } x = 1 \\ [2, 11] & \text{if } x = 2 \end{cases} \]
Labeled all-pairs shortest path (APSP)

- Labeled all pairs shortest path computes these temporal distances, as a function of environment
- Compact encoding using Labeled Value Set (LVS)

\[
\begin{cases}
[6, 11] & \text{if } x = 1 \\
[2, 11] & \text{if } x = 2
\end{cases}
\]
Labeled all-pairs shortest path (APSP)

- Causal link extraction: provides ordering
- In this case, producer *guaranteed* to precede consumer
- Labeled causal link extracted
Labeled Value Set (LVS)

- LVS encodes tightest known value for some condition, as a function of environment
- Ex. Suppose $t < a$ where $a$ depends on environment
- LVS: $t < \{(2, \{x = 1, y = 2\}), (3, \{x = 1\}), (6, \{\})\}$

- **Query** an LVS with $Q$ operator: “what is tightest value over all environments where $x = 1, y = 2$”?  
  - $Q(\{x = 1, y = 2\}) = 2$
  - $Q(\{y = 2\}) = 6$

- **Dominance**: labeled pair $(a_i, \phi_i)$ dominates $(a_j, \phi_j)$ iff $a_i < a_j$ and $\phi_i$ subsumes $\phi_j$

LVS introduced in (Conrad 2009)
Operations on LVS’s

- We use $<$ to compare numbers, but generalizes to other partial order relation $R$
- Operations on LVS’s:
  - Adding new labeled pairs
  - Query
  - Binary operations, like $+$
- See (Conrad 2009) for full details
Labeled all-pairs shortest path (APSP)

- Labeled APSP:
  - Dispatchable form suitable for online execution
  - Allows precedence inference for causal link extraction
  - Detects temporal conflicts

- Computes an LVS for each pair of graph, representing shortest distance as function of environment

- Generalization of Floyd Warshall algorithm that uses LVS operations instead of standard + and <

- See (Conrad 2010) for details
Optimization: Causal link dominance

Concurrent Plan Recognition & Execution for Human-Robot Teams | Steven J. Levine
Slide 59
Optimization: Causal link dominance
Optimization: Causal link dominance

- **Dominance**: Later-occurring producers that are active whenever earlier ones are dominant.

- Reduce number of constraints & solutions
Future Work

• Rank intent/adaptation hypotheses.
  • Probability: consider only likely human intents first

• Richer model for state & human
  • Hybrid state with continuous, spatial variables
  • Hybrid causal links via flow tubes / funnels

• Robot to actively influence human
  • Actively ask clarification questions
  • Informing human of increasingly likely failures (deadlines getting close, likely violated causal link, etc.)
Backup slides
Environments represent sets of subplans

- **Environment**: partial assignment to choice variables
- Represents a *set* of possible subplans
Environments represent sets of subplans

- Ex., \( \{ x_{R1} = \text{juice}, x_{A3} = \text{bagel} \} \) represents:
Environments and subsumption

- Environment $e_1$ subsumes $e_2$ iff $e_2$ contains all assignments in $e_1$
- ex., $\{x_{R1} = juice\}$ subsumes $\{x_{R1} = juice, x_{A3} = bagel\}$
- Intuitively, all subplans represented by $e_2$ also represented by $e_1$ (subset)
Labeled causal links

\[ x = 1 \]
\[ x = 2 \]

\[
\begin{array}{c}
\text{make } p \\
\text{require } p
\end{array}
\]
Labeled causal links

- Label causal links with producer's execution environment
- Ordering determined via labeled APSP
Labeled causal link *dominance*

- **Dominance**: Later-occurring producers with subsuming environments *dominate* others

- Above, later occurring producer dominates earlier one.
Algorithm 8: ExtractCausalLinkConstraints()

Input:
Output:
1. foreach $e_c \in E$ do
2.   foreach $p \in \text{Preconditions}(e_c)$ do
3.     $\xi \leftarrow \text{new LVS \{\}}$ with relation $\prec$
4.     foreach $e \neq e_c \in E$ with $p$ or $\neg p$ in $\text{Effects}(e)$ do
5.       if not $e_c \prec e$ and $\phi_e \land \phi_{e_c}$ is feasible(!) then
6.         AddLVS$((e, \phi_e), \xi)$
7.     end
8.     Separate $\xi$ into $P$ and $T$
9.     Create decision variable $a_{e_c, p}$ with domain $P$
10.    AddConstraint$\left(\phi_{e_c} \Rightarrow \bigvee_i (a_{e_c, p} = e_p) \land \phi_{e_c} \land \phi_{e_c} \right)$
11.    foreach $e_{p_i} \in P$ where not $e_{p_i} \prec e_c$ do
12.       Add $[\epsilon, \infty]$: $\{a_{e_{p_i}, p} = e_{p_i} \land \phi_{e_{p_i}} \land \phi_{e_{p_i}} \text{ from } e_{p_i} \text{ to } e_c\}$
13.    end
14.    foreach $e_{p_i} \in P$ do
15.      foreach $e_{t_i} \in T$ do
16.         $\phi_C \leftarrow \{a_{e_{p_i}, e} = e_{p_i} \land \phi_{e_{p_i}} \land \phi_{e_{c, p}} \land \phi_{e_{c}} \land \phi_{e_{c}}\}$
17.         if $e_{p_i} \prec e_{t_i} \text{ and } e_{t_i} \prec e_{c} \text{ and } e_{c} \prec e_{c}$ then
18.             AddConstraint$(-\phi_C)$
19.         else if $e_{p_i} \prec e_{t_i} \text{ and } e_{t_i} \parallel e_{c} \text{ and } e_{c} \parallel e_{c}$ then
20.             Add $[\epsilon, \infty]$: $\phi_{e_{t_i}}$ from $e_{p_i}$ to $e_{t_i}$
21.         else if $e_{p_i} \parallel e_{t_i} \text{ and } e_{t_i} \prec e_{c} \text{ and } e_{c} \parallel e_{c}$ then
22.             Add $[\epsilon, \infty]$: $\phi_{e_{c}}$ from $e_{p_i}$ to $e_{t_i}$
23.         else if $e_{p_i} \parallel e_{t_i} \text{ and } e_{t_i} \parallel e_{c} \text{ and } e_{c} \parallel e_{c}$ then
24.             Create decision variable $b_{e_{p_i}, e_{c, p}, e_{c, t_i}}$ with domain $\{1, 2\}$
25.             Add $[\epsilon, \infty]$: $\{b_{e_{p_i}, e_{c, p}, e_{c, t_i}} = 1\} \land \phi_{e_{c}}$ from $e_{t_i}$ to $e_{p_i}$
26.             Add $[\epsilon, \infty]$: $\{b_{e_{p_i}, e_{c, p}, e_{c, t_i}} = 2\} \land \phi_{e_{c}}$ from $e_{c}$ to $e_{t_i}$
27.         end
28.     end
29. end
30. end
31. end
32. end
TPN Encodings

• What useful things can be encoded by TPN’s?
  • Resource / agent allocation
  • Recovery actions
  • Flexibility to execute different tasks (HTN-like)
Random TPNU generation

- Random sequential, parallel, and choice structure
- Randomly-generated causal link structure (encoded through plant domain) with potential threats
- Temporal “squeezing”
- Wide range of problem sizes
Labeled causal link extraction

- Uses an LVS, except:
  - Values are TPNU events, rather than numbers
  - Relation $R$ is not $<$ but rather succession (via labeled APSP):
    $e_a \mathrel{R} e_b = \begin{cases} \text{TRUE} & \text{if } Q_{de_a \rightarrow e_b} (\phi_a \cup \phi_b) \leq 0 \\ \text{FALSE} & \text{otherwise} \end{cases}$

- To find producers for consumer ec requiring $p$:
  - Insert all $e_p$ that produce $p$ or $\neg p$ into LVS
  - Extract threat resolution constraints from those producing $\neg p$
  - Extract labeled causal links from those producing $p$
  - (See paper for full details)
Encoding in an ATMS

A. \[ z = 1 \Rightarrow (a_c = 1 \lor (a_c = 2 \land y = 1)) \]

\[ \neg (a_c = 1 \land x = 2 \land z = 1) \]

B. \[
\begin{align*}
\phi_1 & \quad \text{Holds} \\
\phi_2 & \quad \text{Holds} \\
\phi_3 & \quad \text{Links} \\
\phi_4 & \quad \text{L} \\
\end{align*}
\]

Consistent

\[
\begin{align*}
\{ z = 1 \Rightarrow (a_c = 1 \lor (a_c = 2 \land y = 1)) \} \\
\{ \neg (a_c = 1 \land x = 2 \land z = 1) \} \\
\end{align*}
\]
A basic threat
A basic threat

\[ x = 1 \]

\[ x = 2 \]

\[ y = 1 \]

\[ y = 2 \]

\[ z = 1 \]

\[ z = 2 \]

\[ p : \{y = 1\} \]

\[ p : \{\} \]

\[ \text{make } p \]

\[ \text{not } p \]

\[ \text{require } p \]
A basic threat

\[ a_c = \begin{cases} 
1 & \text{if first causal link enforced} \\
2 & \text{if second causal link enforced} 
\end{cases} \]
Extracting propositional constraints

\[ z = 1 \implies (a_c = 1 \lor (a_c = 2 \land y = 1)) \]
\[ \neg (a_c = 1 \land x = 2 \land z = 1) \]

At least 1 causal link holds

Threat resolved
Causal link extraction in a nutshell*

- For each precondition of each **consumer** event:
  - Find all **producer** provably before or during consumer
Causal link extraction in a nutshell*

• For each precondition of each **consumer** event:
  • Find all **producer** provably before or during consumer
  • Add propositional & temporal constraints for each producer