Concurrent Plan Recognition & Execution for Human-Robot Teams

Cognitive Robotics 2016 Lecture

Steven J. Levine

Wednesday, March 16th, 2016

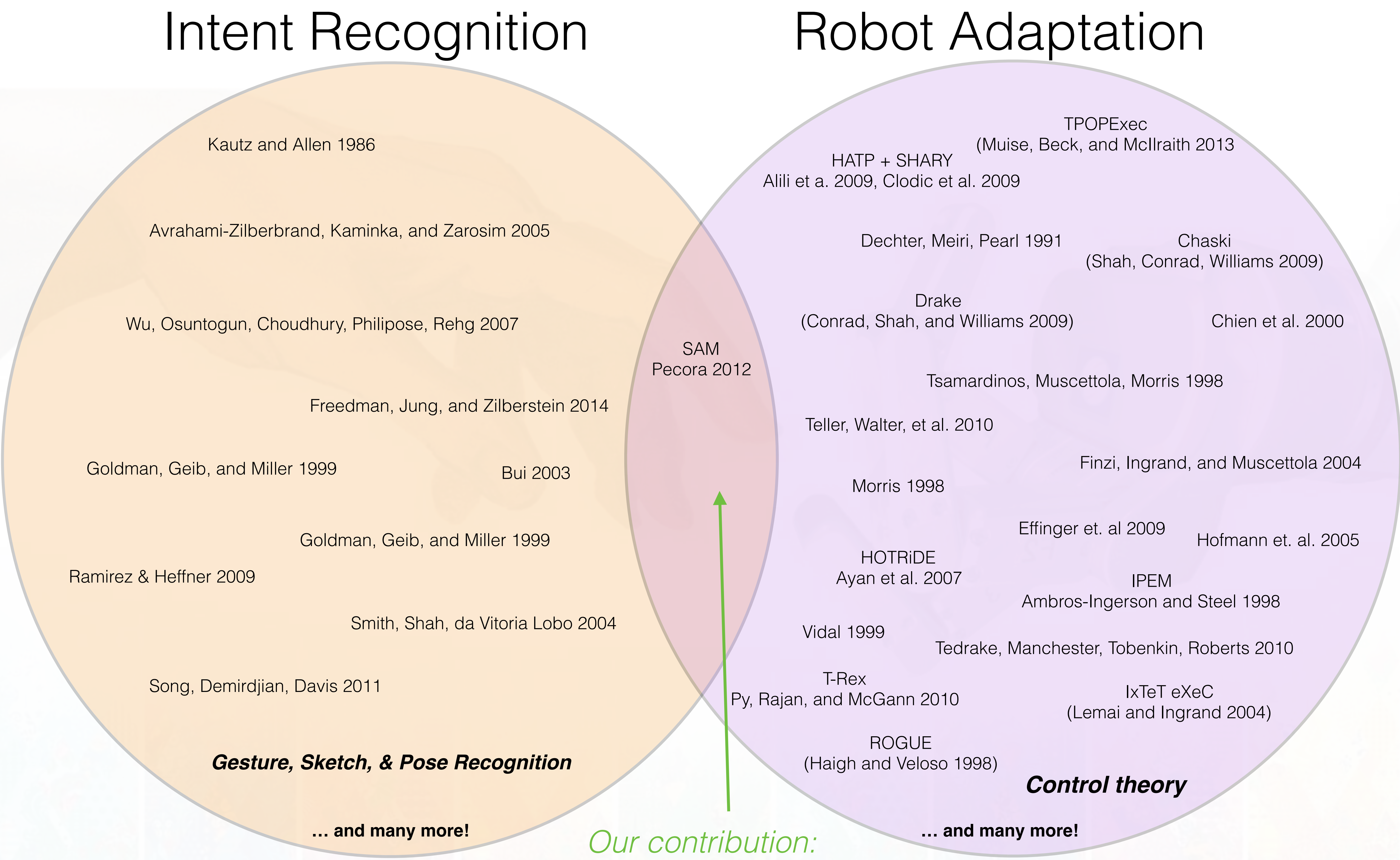
Intent recognition & adaptation are siblings

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

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



Our contribution:

Pike

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

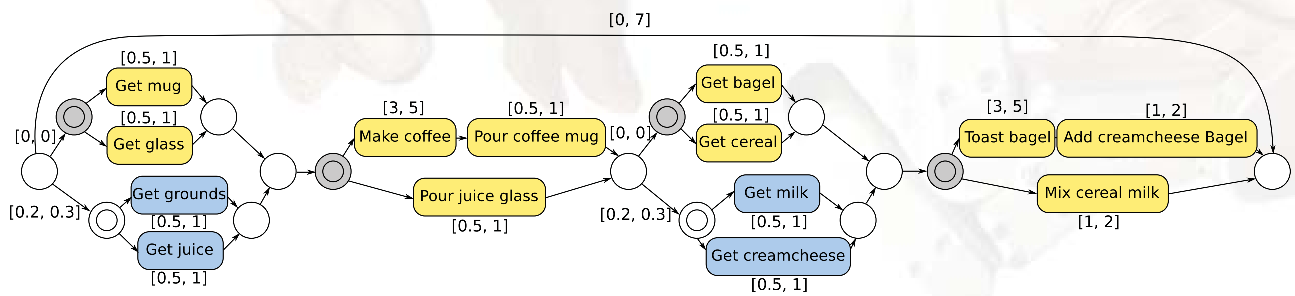
How to recognize intent & adapt?

$$\left\{\begin{array}{c}\text{Intent recognition}\\ \text{Robot adaptation}\end{array}\right\} \text{is} \left\{\begin{array}{c}\text{recognizing}\\ \text{making}\end{array}\right\} \text{decisions consistent with team's task goals}$$

- Assume rational, cooperative agents
- Prune any (irrational) decisions resulting in plan failure:
  - Unmet action preconditions: ⇒ Causal link reasoning
  - Missed deadlines: ⇒ Temporal conflicts
  - Unanticipated failures: ⇒ Online execution monitoring

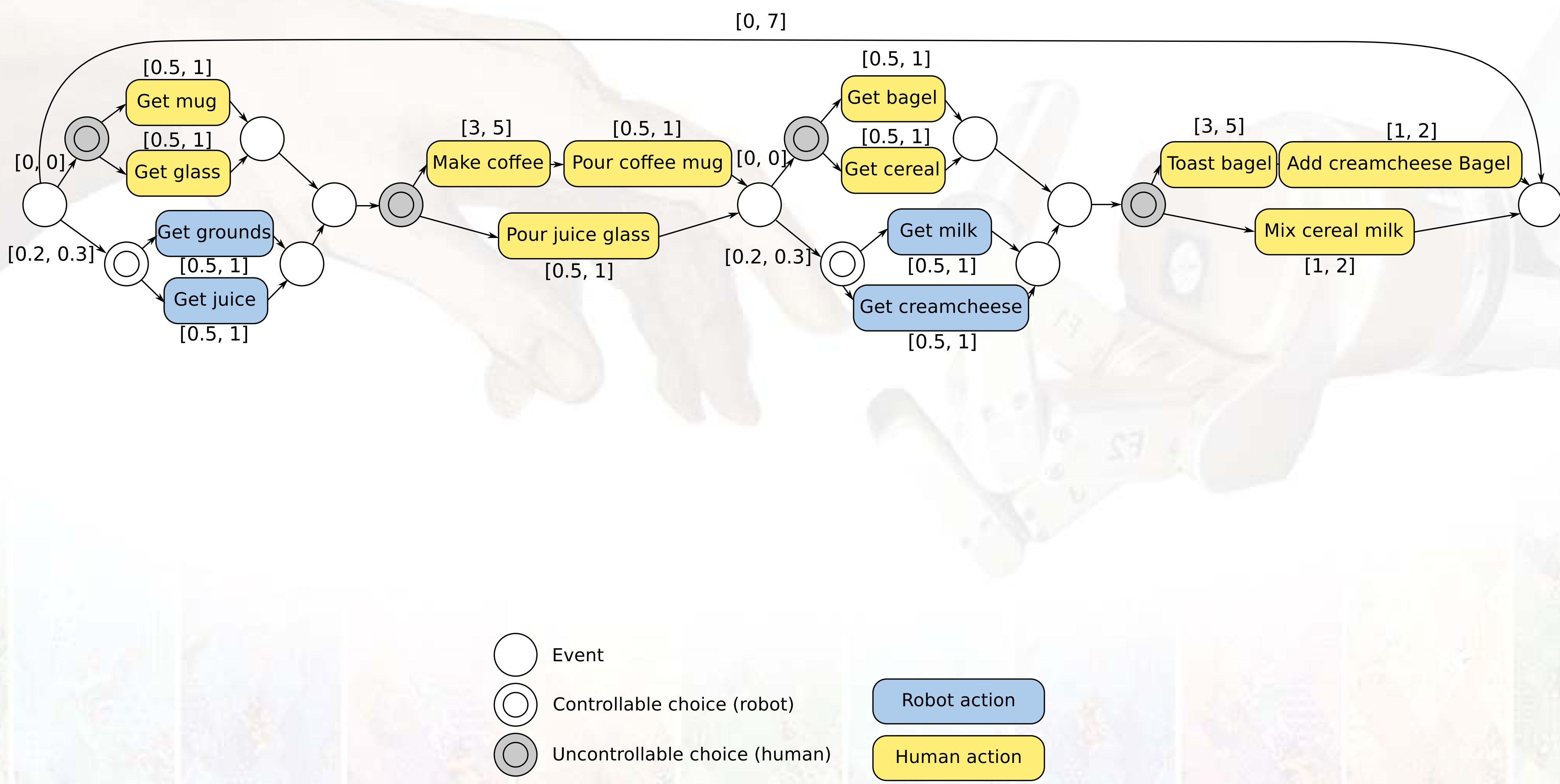
Approach in a nutshell

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



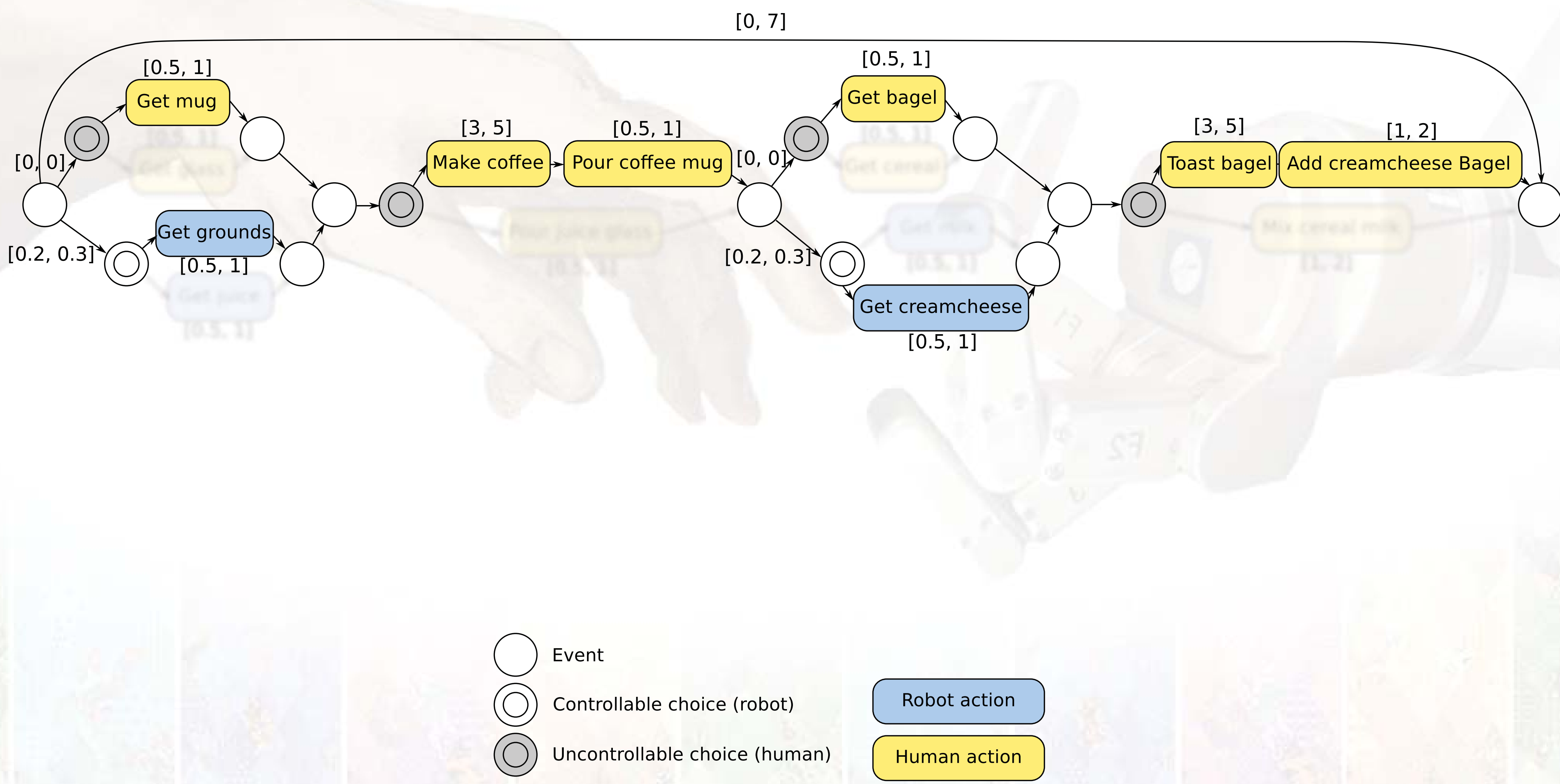
Contingent, temporally-flexible plans

Temporal Planning Network with Uncertainty (TPNU)



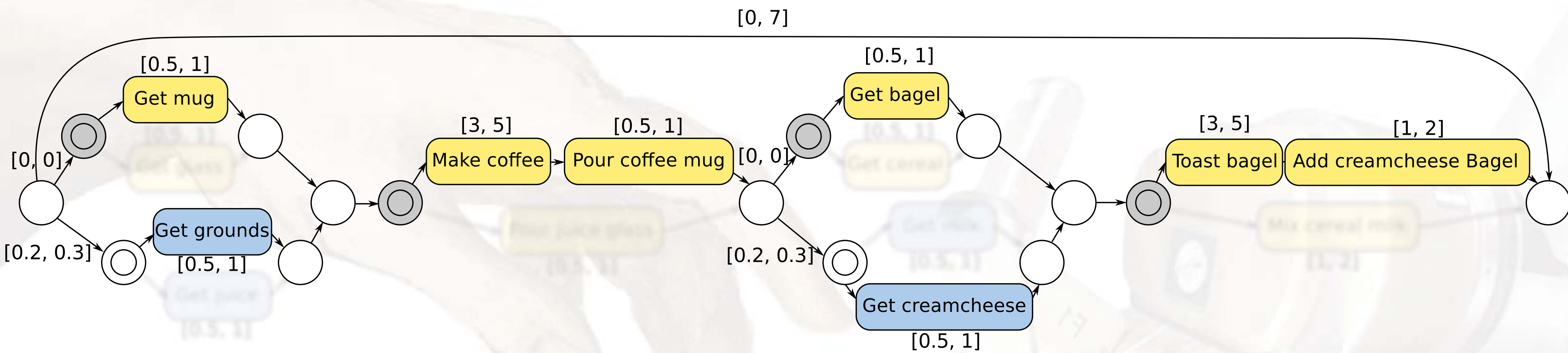
Contingent, temporally-flexible plans

Temporal Planning Network with Uncertainty (TPNU)



Contingent, temporally-flexible plans

Temporal Planning Network with Uncertainty (TPNU)

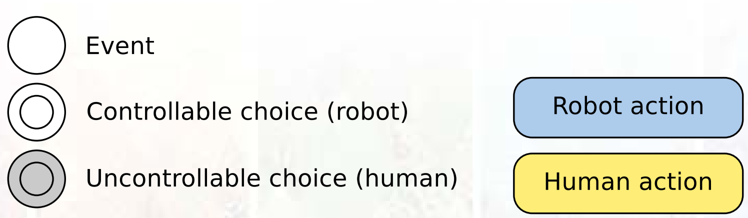


This candidate subplan is temporally inconsistent.

Conflict:

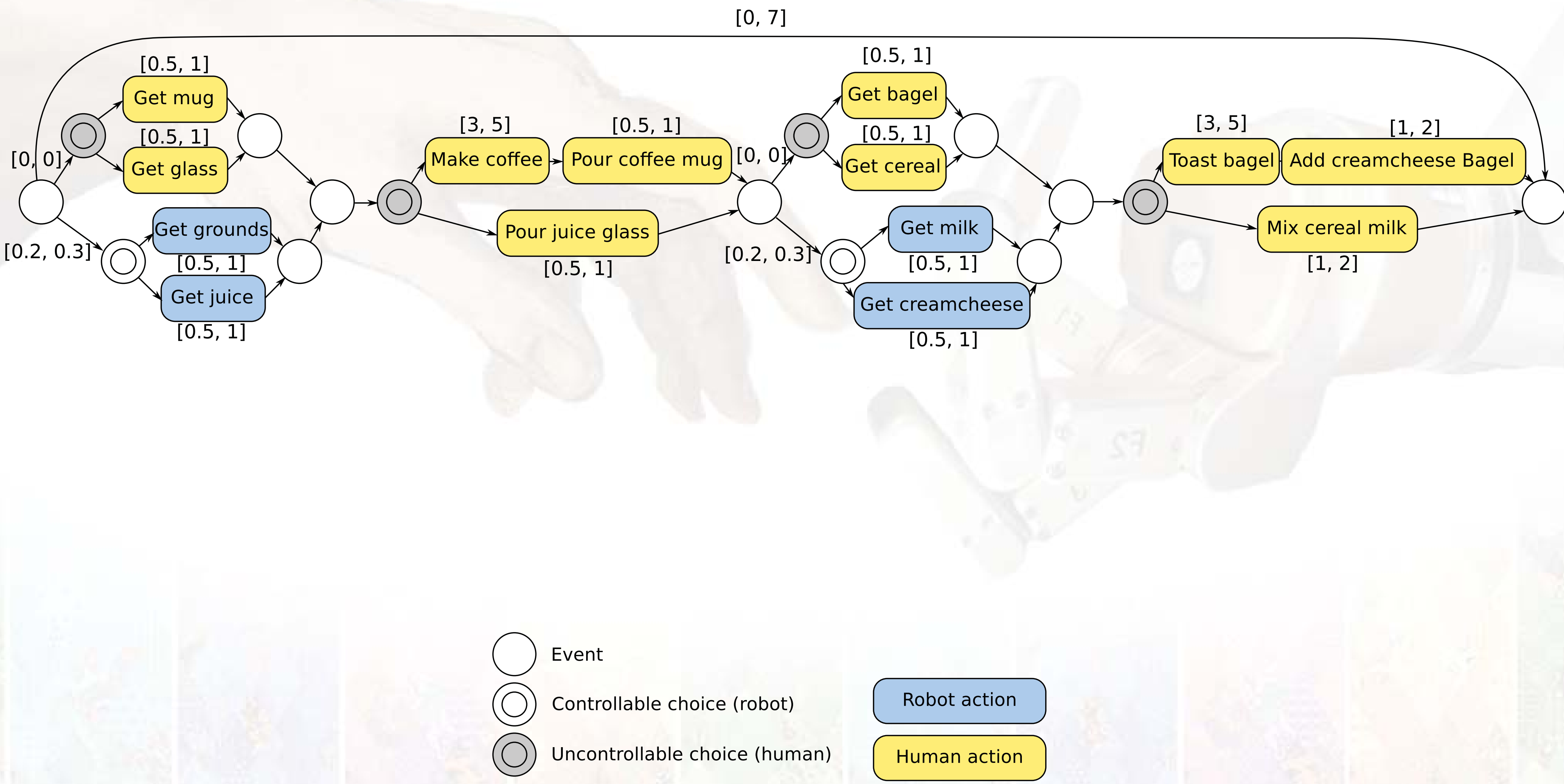
(Conrad 2009)

$$\neg(x_{A_2} = coffee \land x_{A_4} = bagel)$$

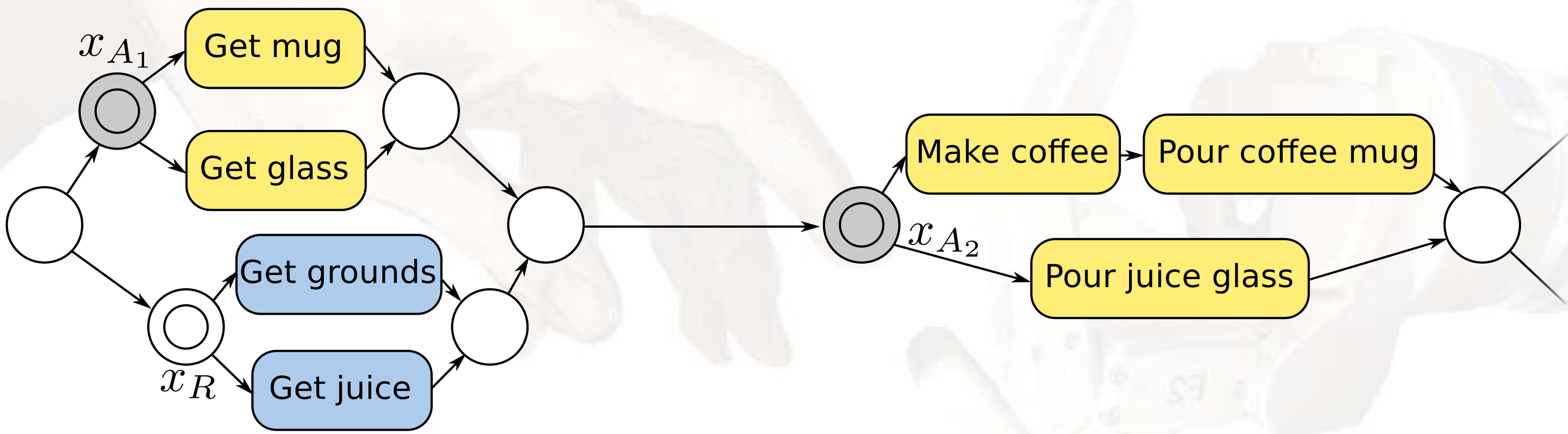


Contingent, temporally-flexible plans

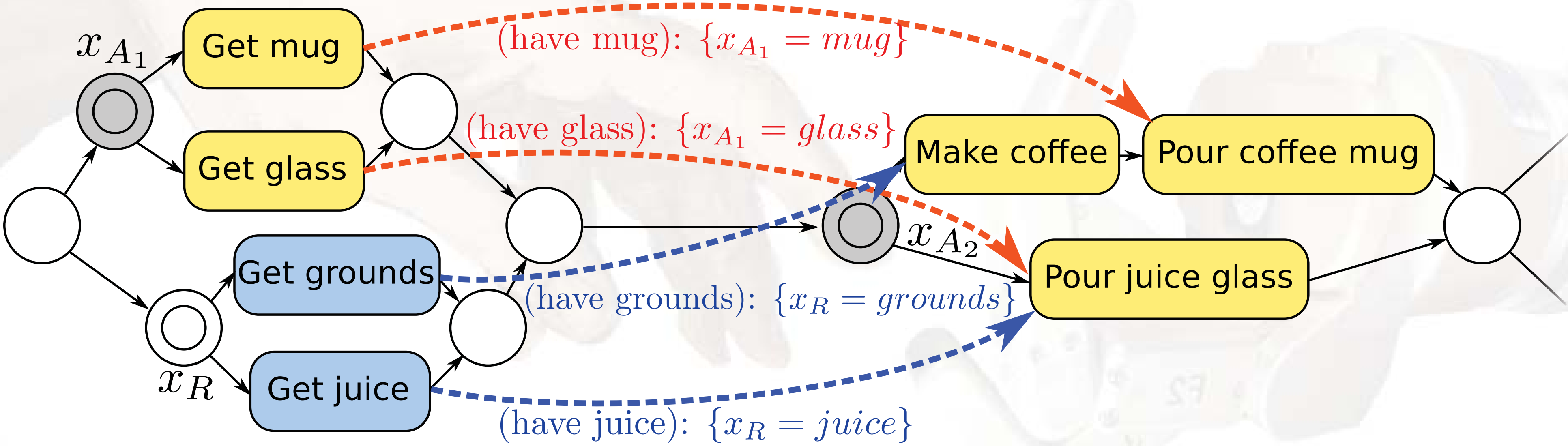
Temporal Planning Network with Uncertainty (TPNU)



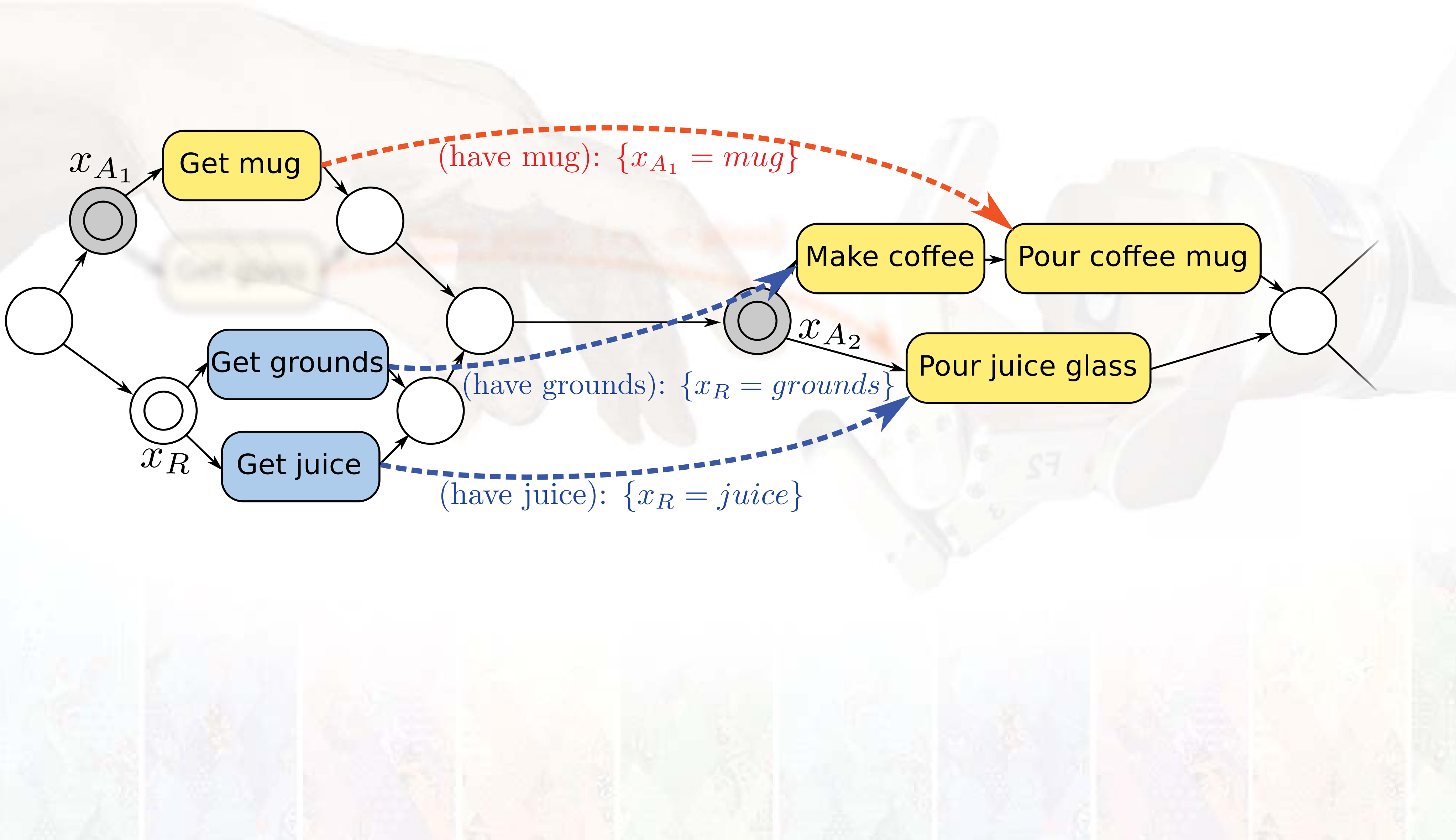
First part: making a drink



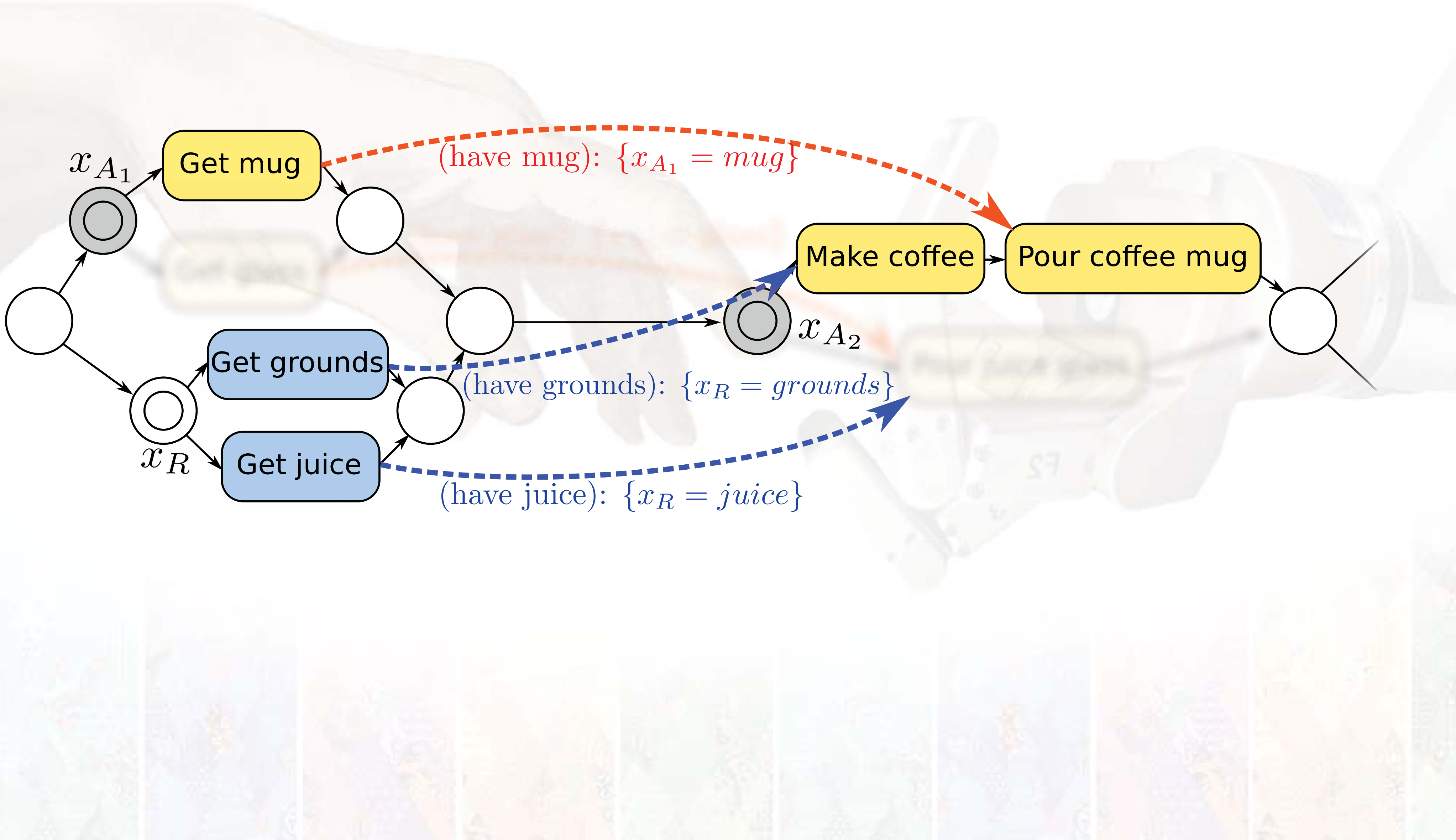
Extracting labeled causal links



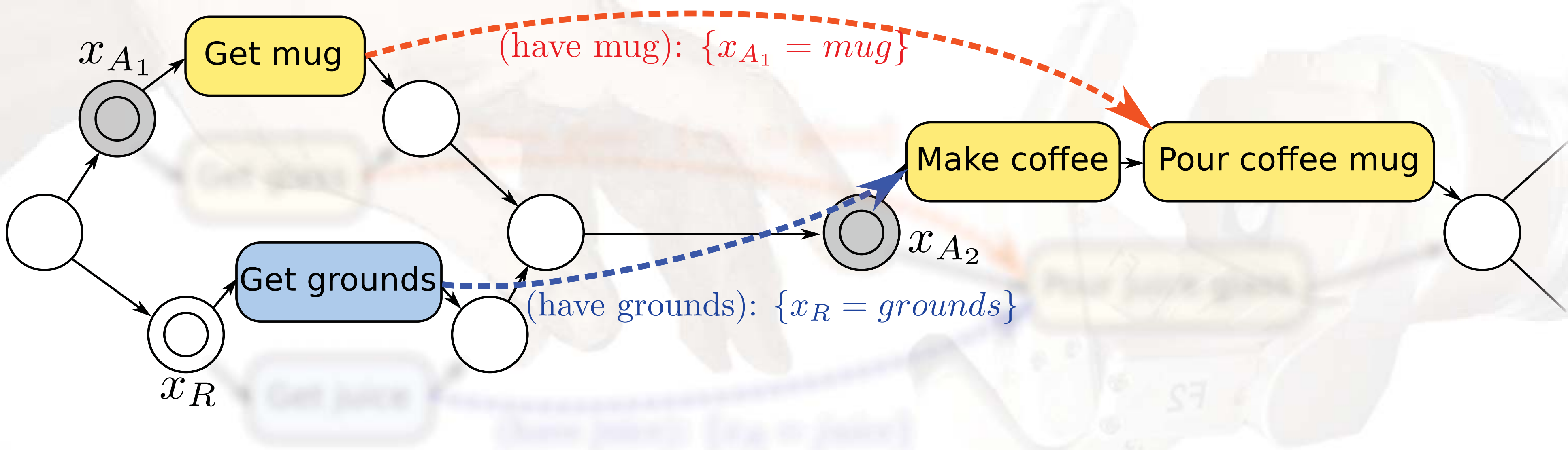
Suppose person picks up mug…



…so can't pour juice later…



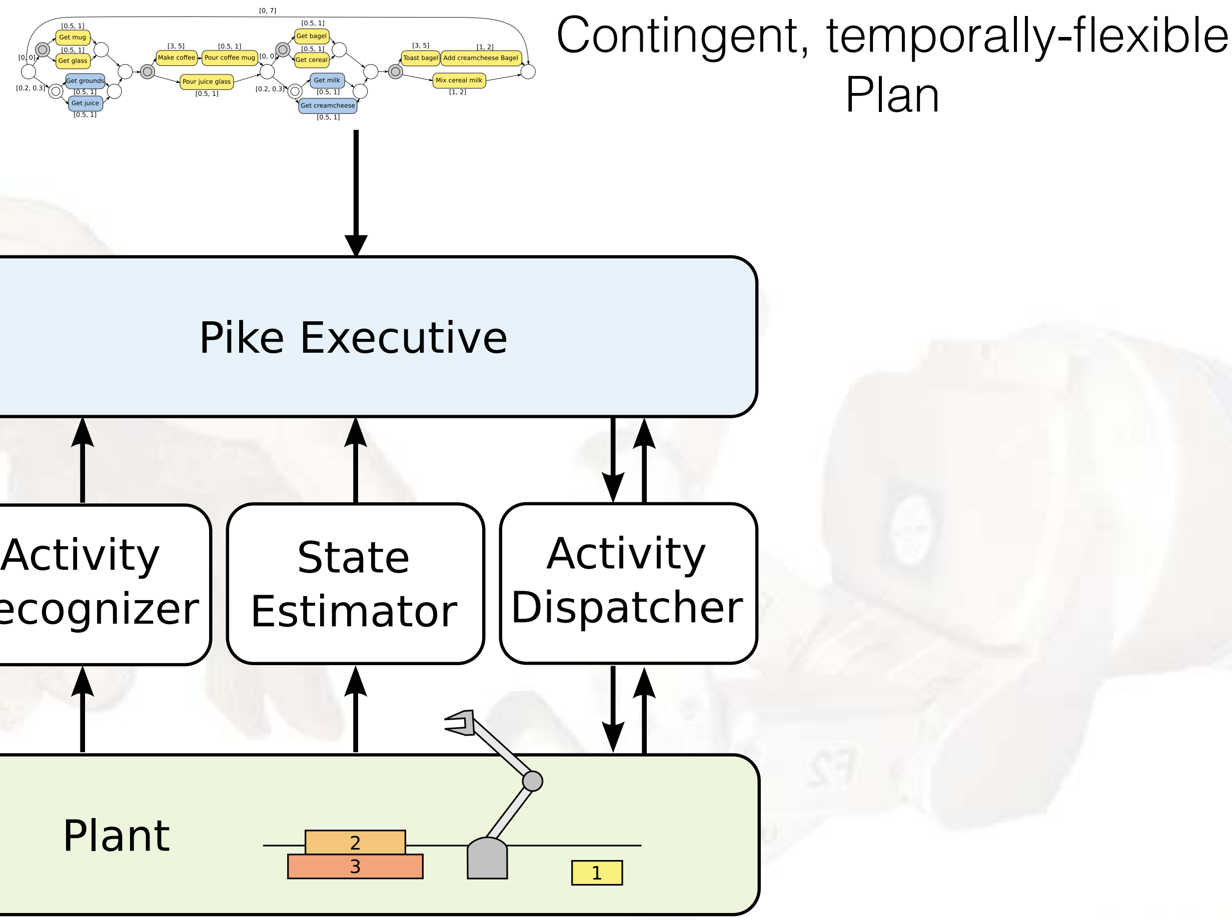
…so robot should get coffee now.



- **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

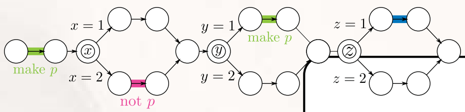
Pike in larger architecture



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

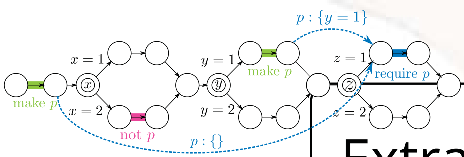
Offline

Online



Labeled APSP

⇓

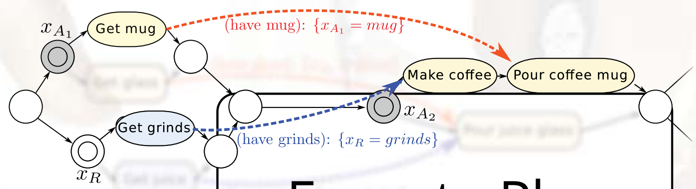


Extract Labeled Causal Links

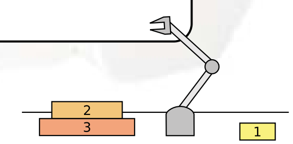
⇓

$$\begin{cases} z=1 \Rightarrow (a_c=1 \lor (a_c=2 \land y=1)) \\ \neg(a_c=1 \land x=2 \land z=1) \end{cases}$$

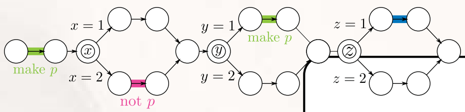
Compile Constraints



Execute Plan

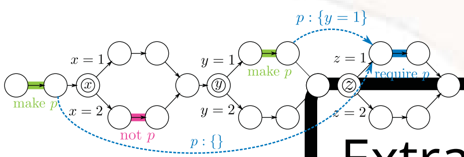


Offline



Labeled APSP

⇓



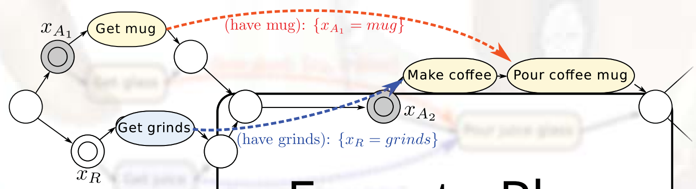
Extract Labeled Causal Links

⇓

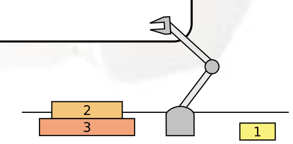
$$\begin{cases} z = 1 \Rightarrow (a_c = 1 \vee (a_c = 2 \wedge y = 1)) \\ \neg(a_c = 1 \wedge x = 2 \wedge z = 1) \end{cases}$$

Compile Constraints

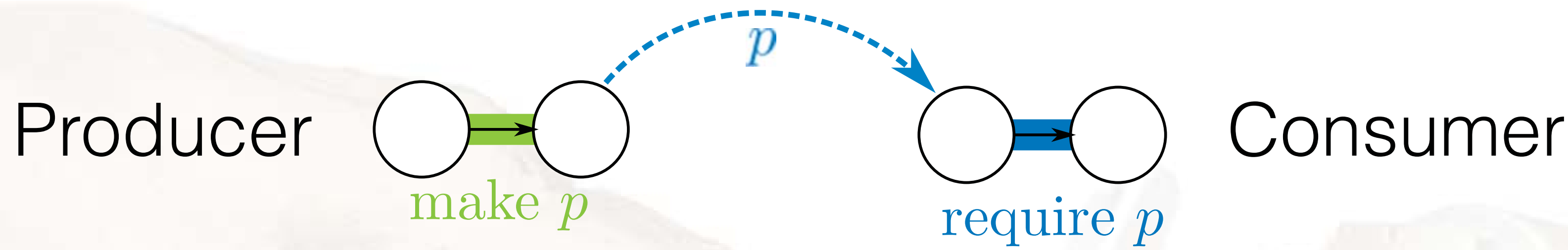
Online



Execute Plan

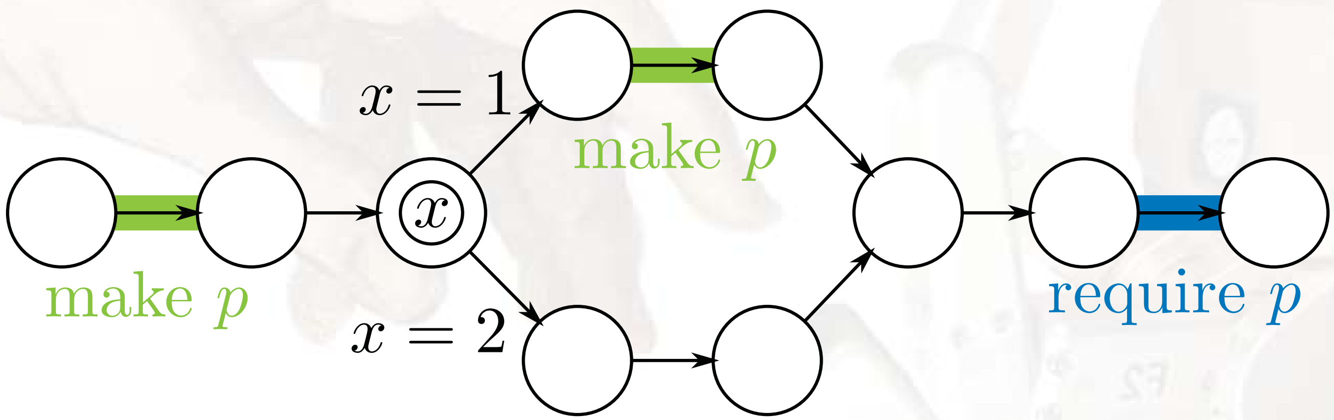


Causal links justify action preconditions

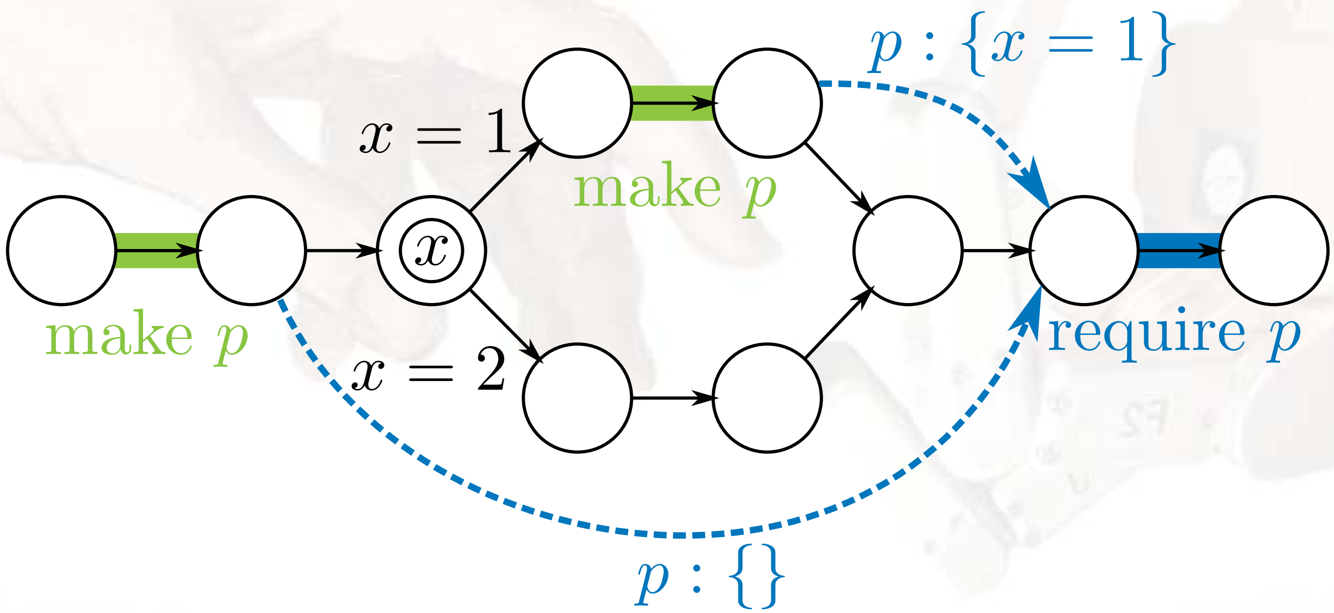


- Insufficient for contingent, temporally-flexible plans:
  - What if producer doesn't execute?
  - What if consumer doesn't execute?
  - Determining ordering is non-trivial
- We generalize to ***labeled causal links***
  - Encode requisite choices for causal link to hold

Labeled causal links

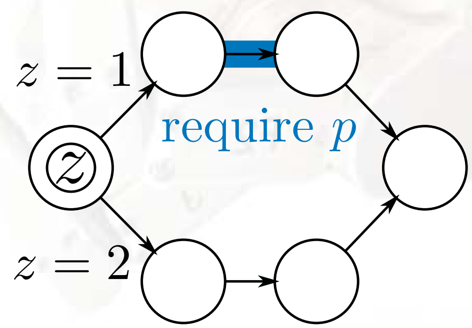


Labeled causal links



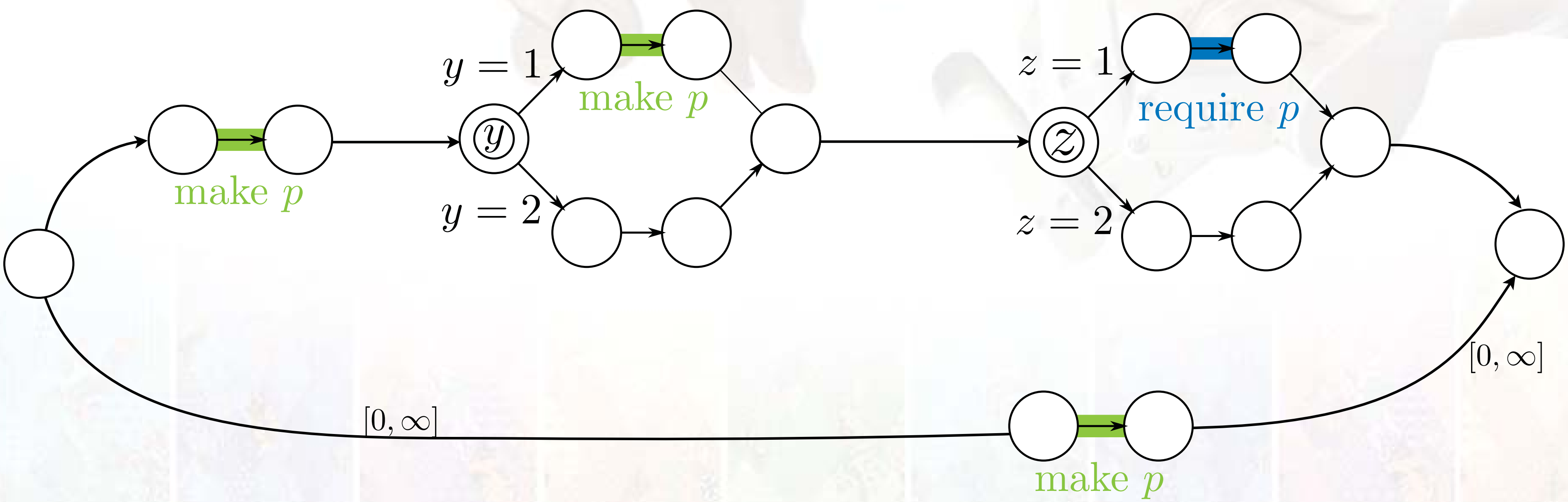
Causal link extraction in a nutshell*

- For each precondition of each **consumer** event:



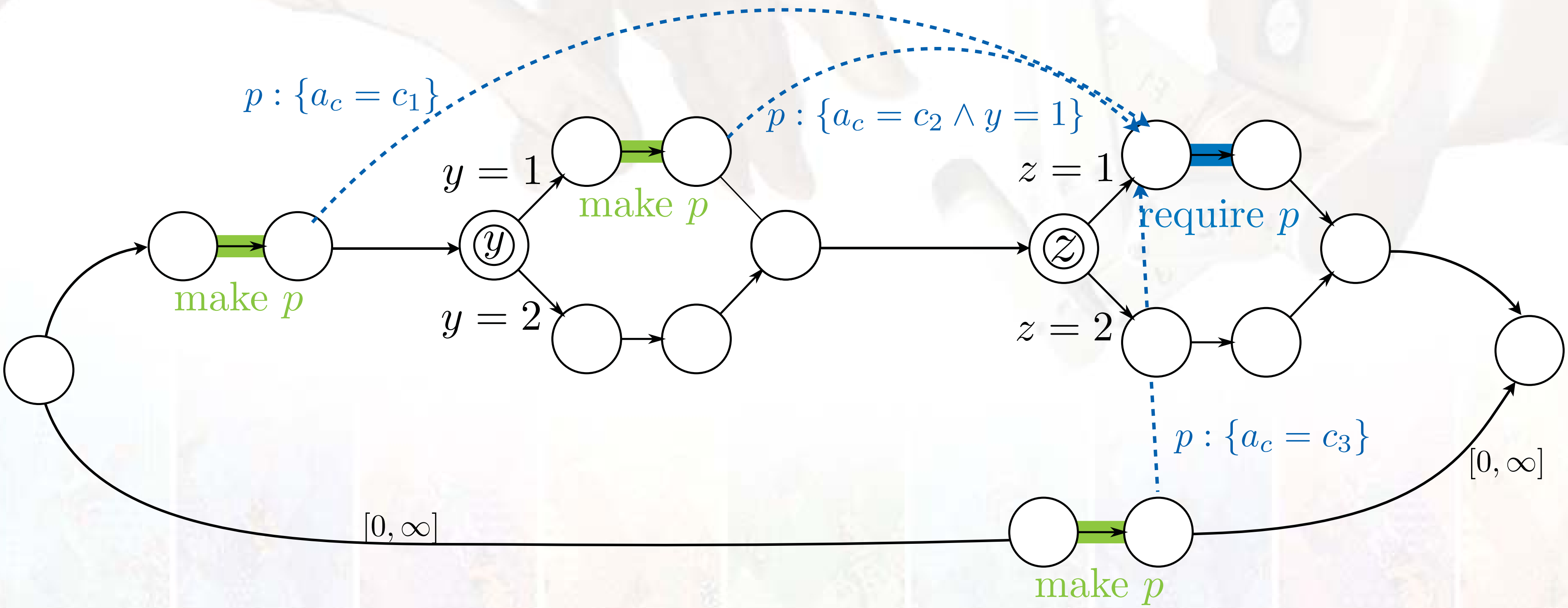
Causal link extraction in a nutshell*

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



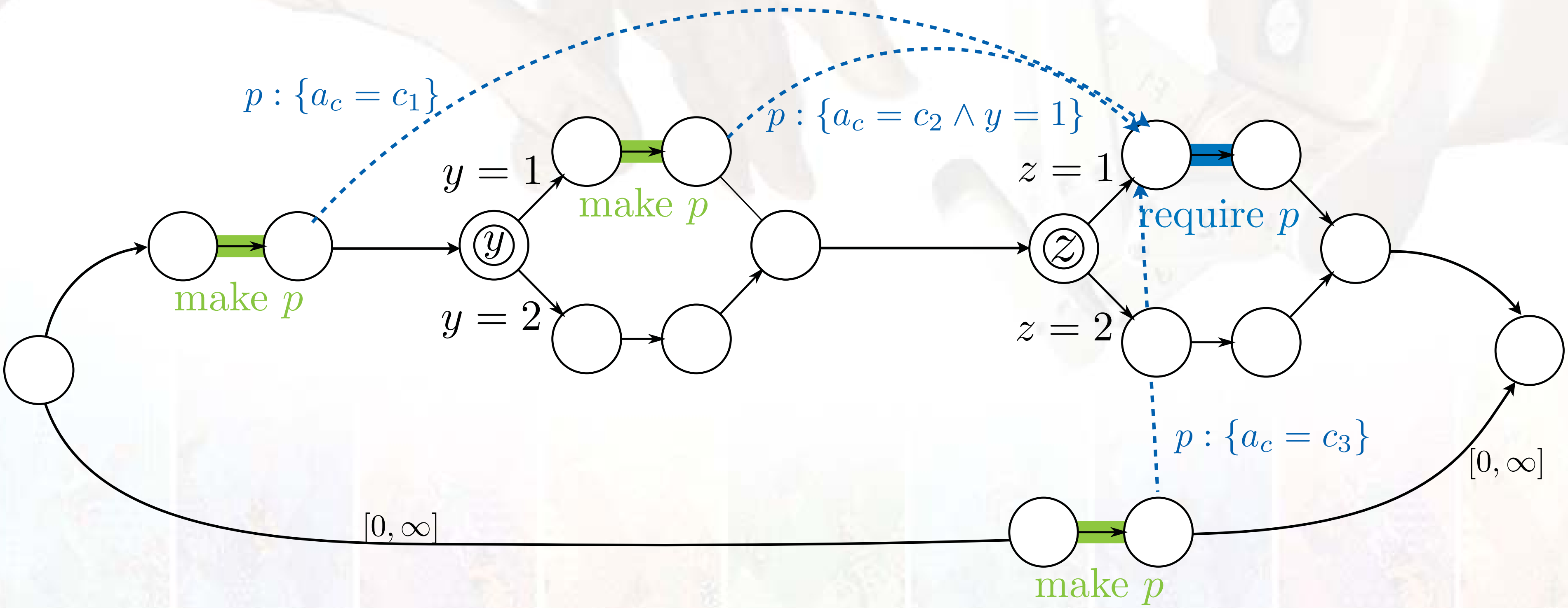
Causal link extraction in a nutshell*

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



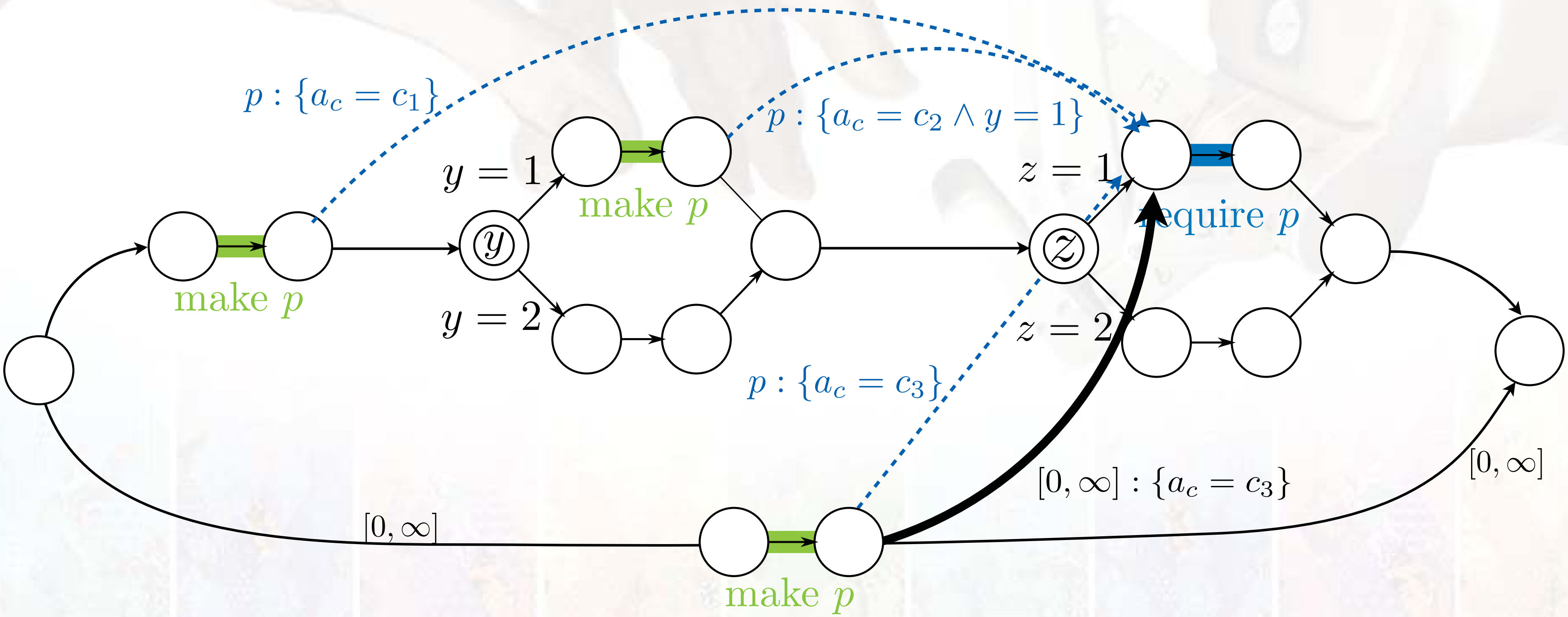
Causal link extraction in a nutshell*

- 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*

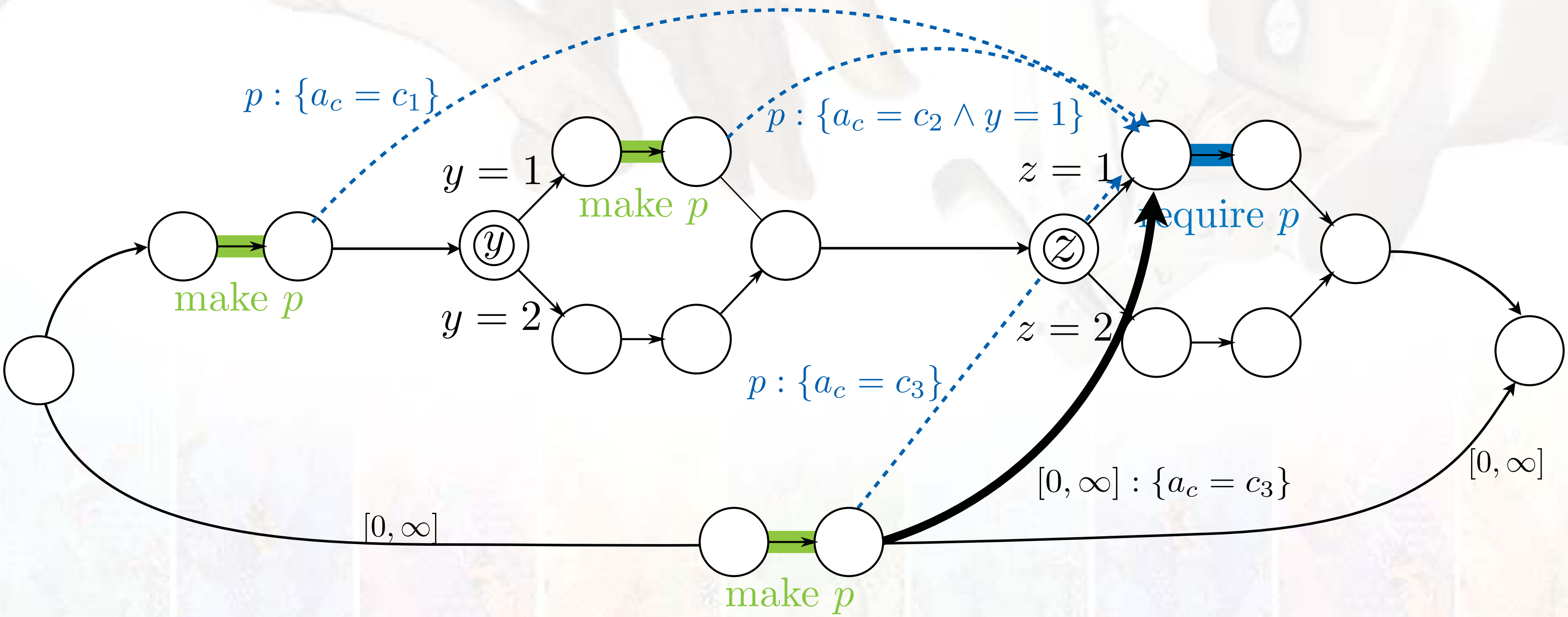
- For each precondition of each **consumer** event:
  - Find all **producers** provably before or during consumer
    - Add propositional & temporal constraints for each producer



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: $(z = 1) \Rightarrow \left[(a_c = c_1) \vee (a_c = c_2 \wedge y = 1) \vee (a_c = c_3) \right]$

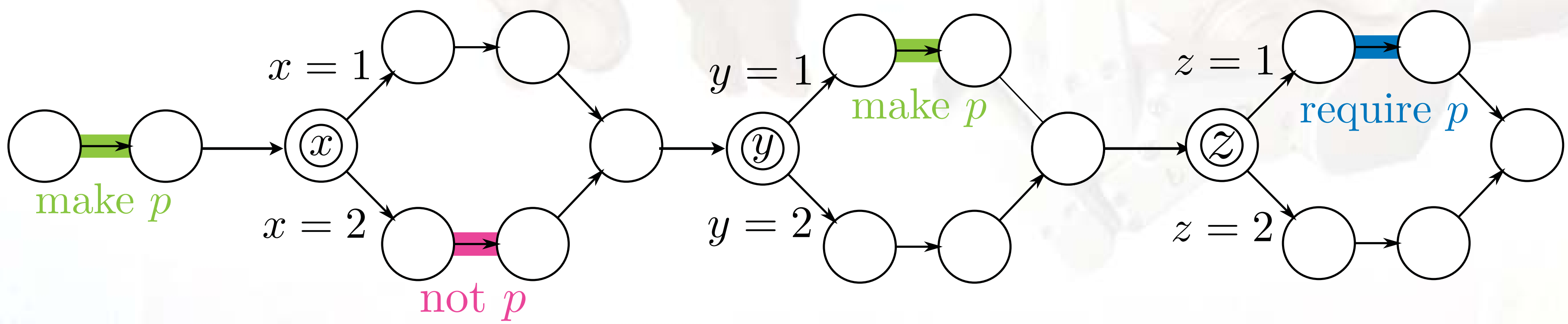


Causal link extraction in a nutshell*

- 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

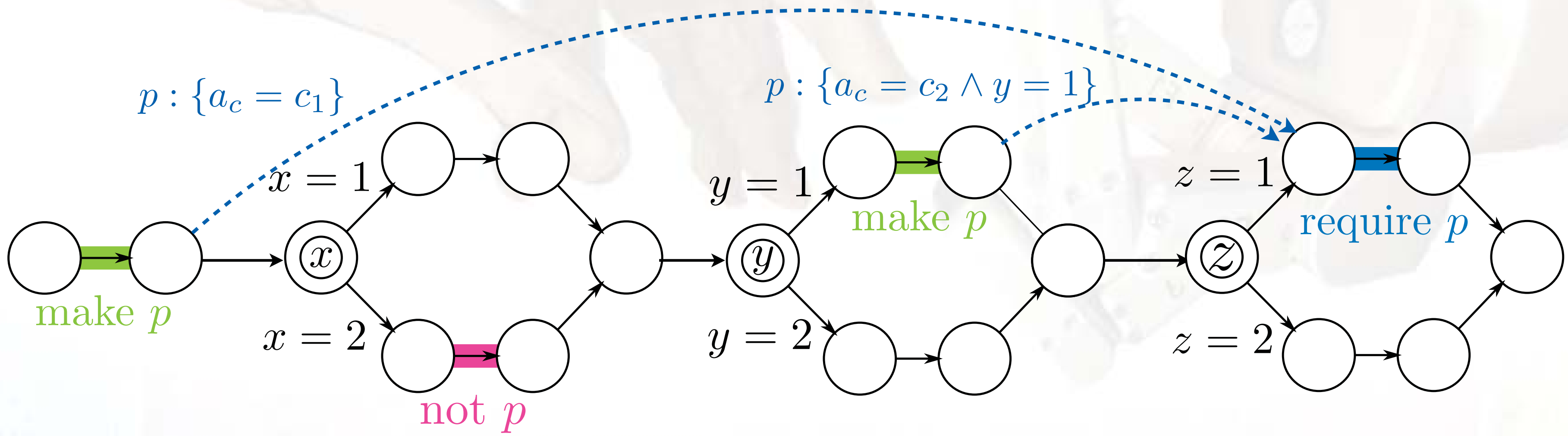
Causal link extraction in a nutshell*

- 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



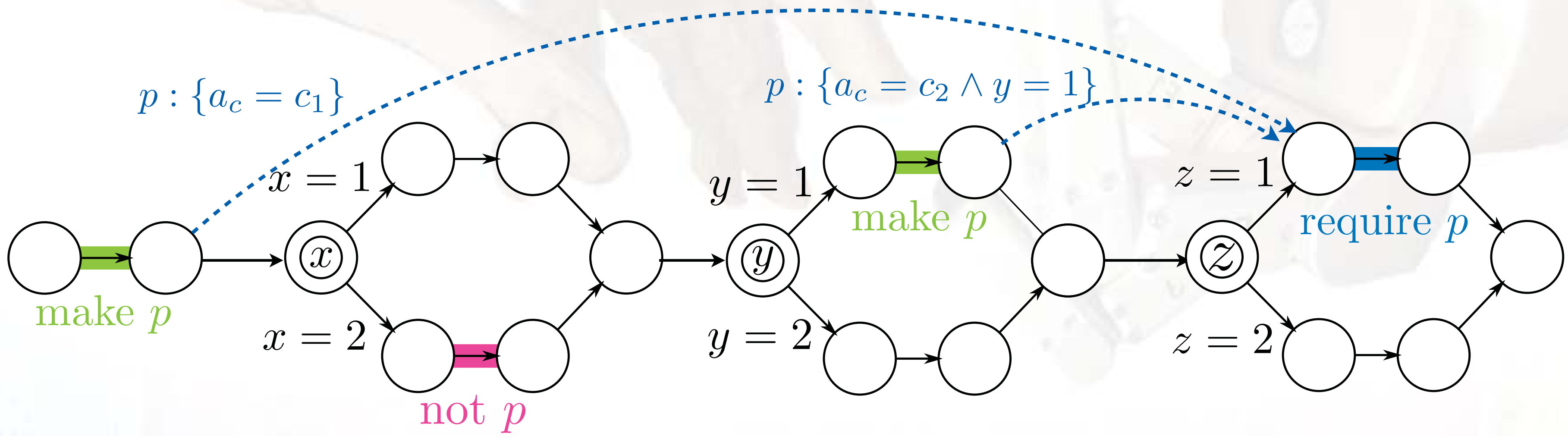
Causal link extraction in a nutshell*

- 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*

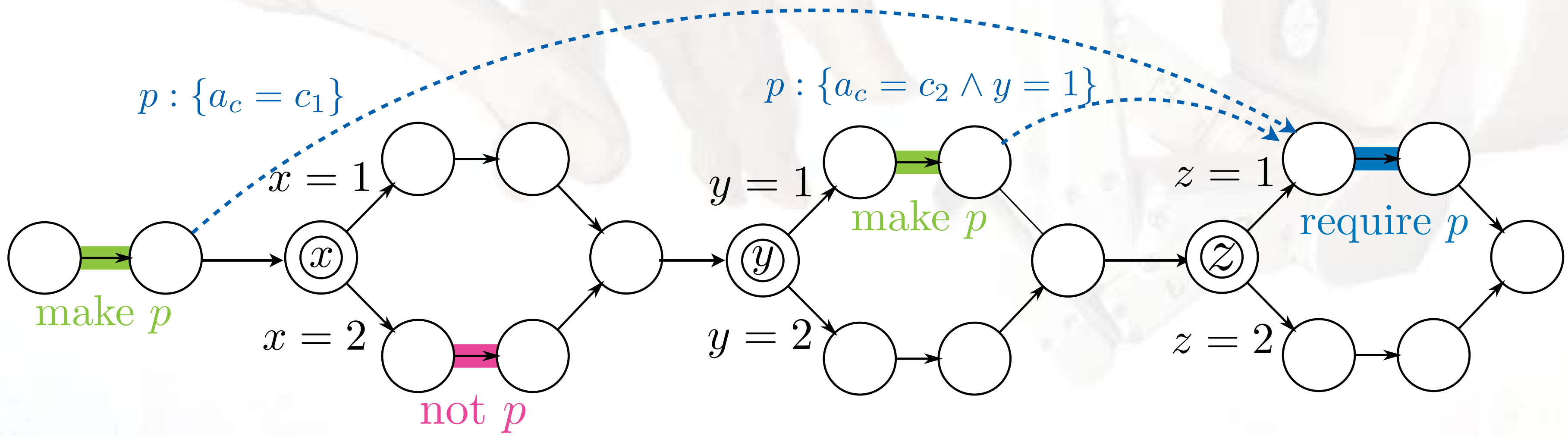
- For each precondition of each **consumer** event:
  - Find all **producers** provably before or during consumer
    - Add propositional & temporal constraints for each producer
  - Find all potential threats probably before or during causal link



$$(z = 1) \Rightarrow \left[(a_c = c_1) \vee (a_c = c_2 \wedge y = 1)\right]$$

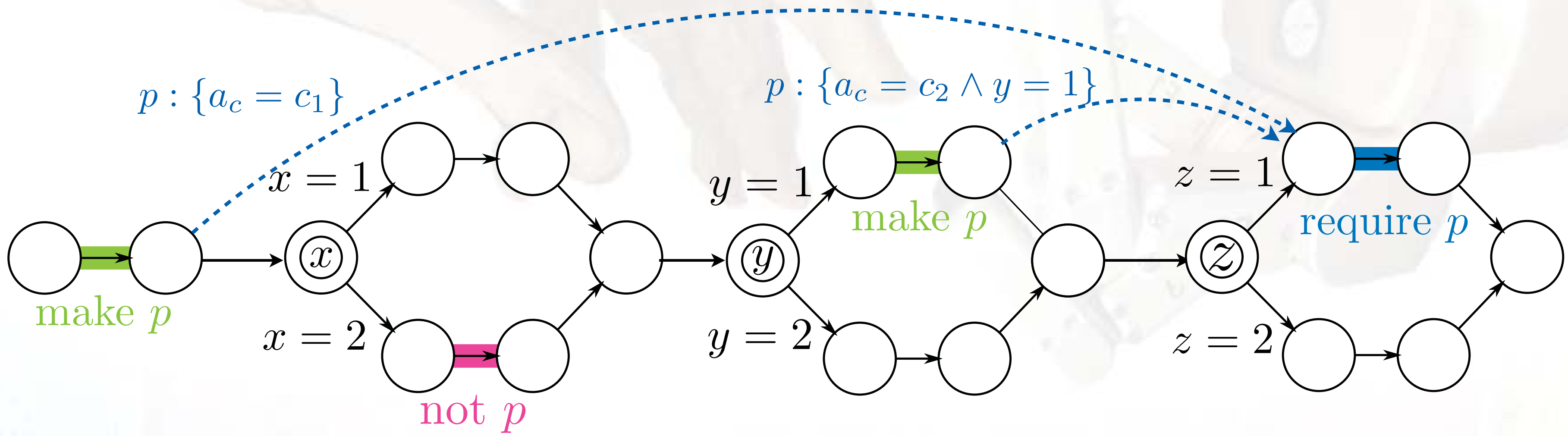
Causal link extraction in a nutshell*

- 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*

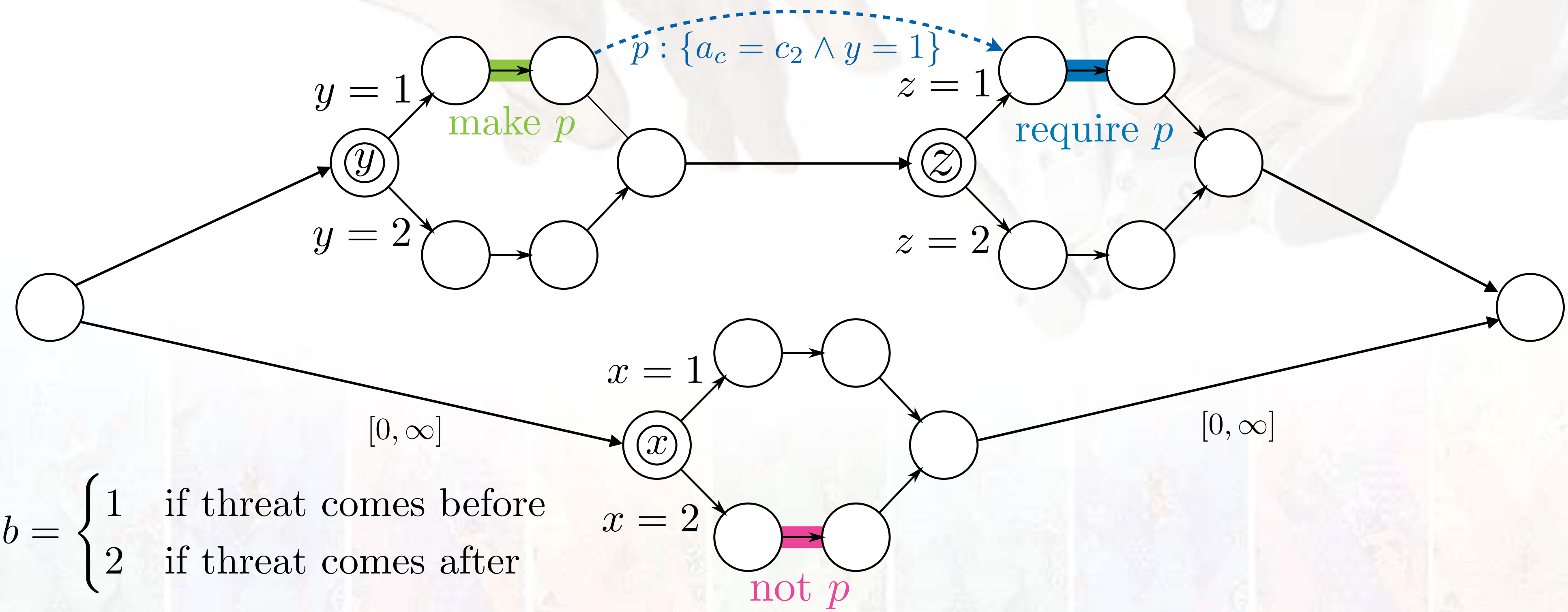
- For each precondition of each **consumer** event:
  - Find all **producers** provably before or during consumer
    - Add propositional & temporal constraints for each producer
  - Find all potential threats probably before or during causal link
    - Resolve via additional propositional & temporal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



Constraint: $\neg(z = 1 \wedge a_c = c_1 \wedge x = 2)$

Causal link extraction in a nutshell*

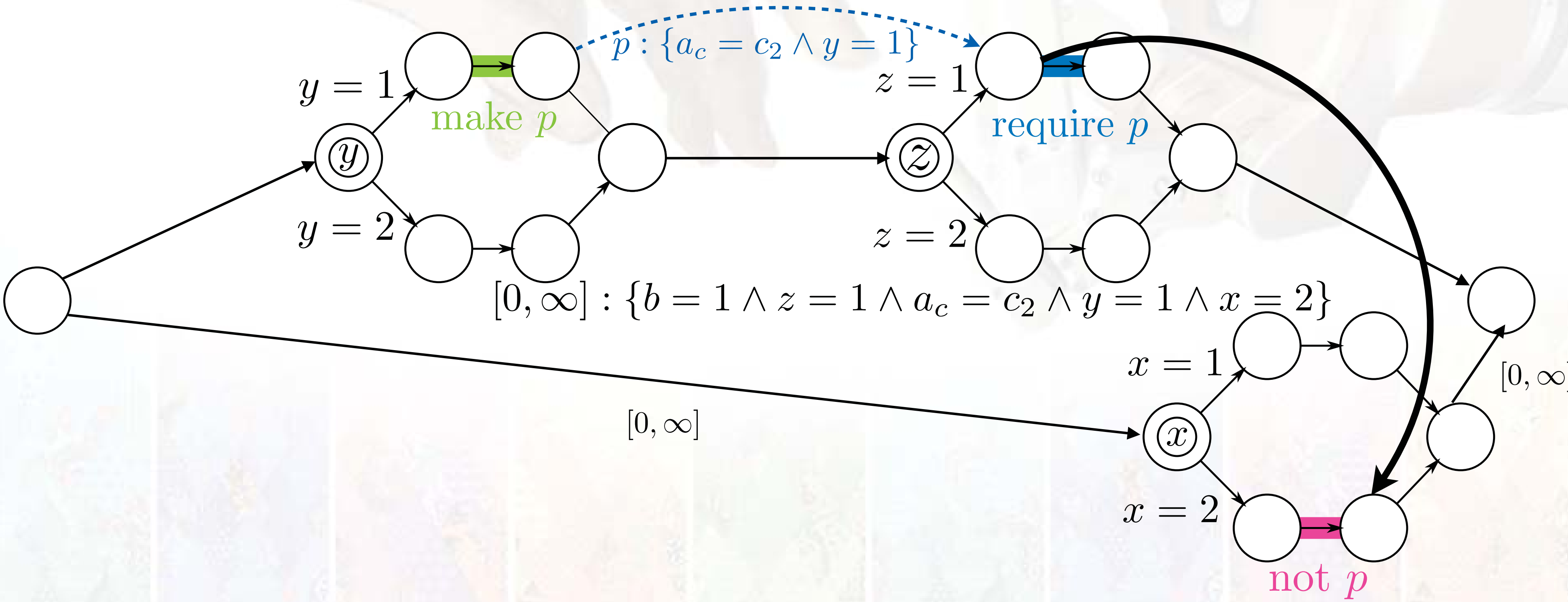
- For each precondition of each **consumer** event:
  - Find all **producers** provably before or during consumer
    - Add propositional & temporal constraints for each producer
  - Find all potential threats probably before or during causal link
    - Resolve via additional propositional & temporal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



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

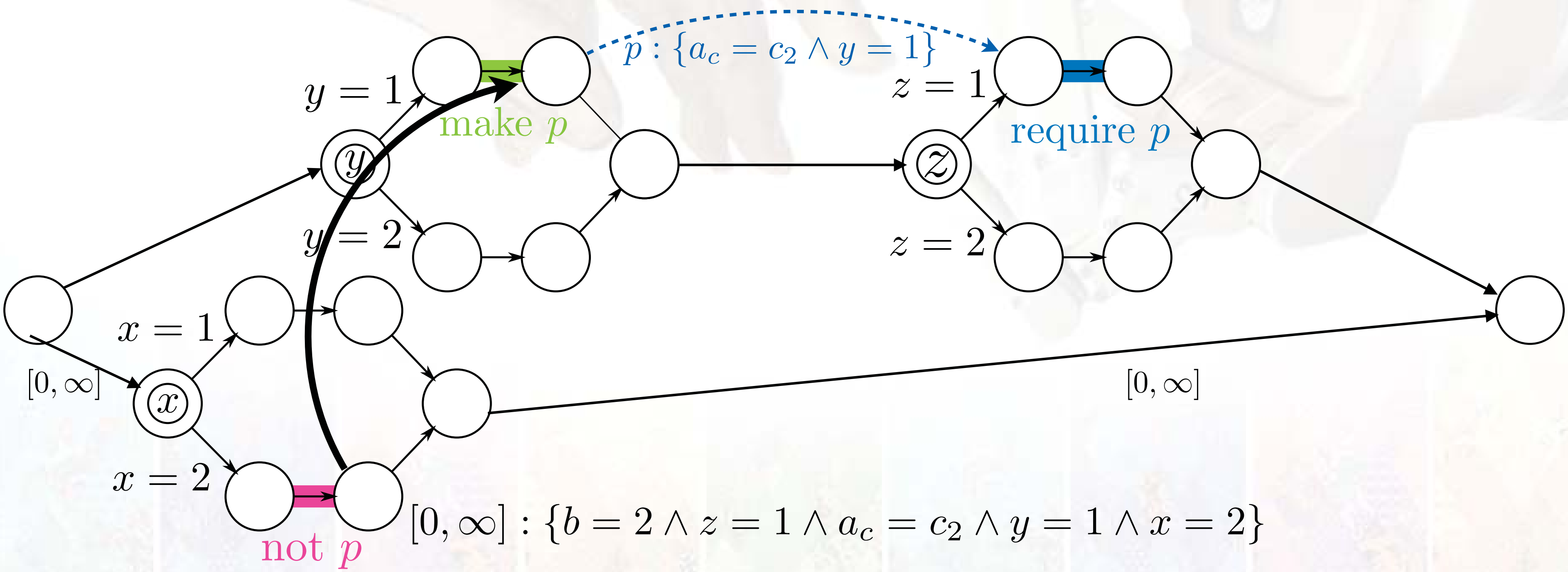
Causal link extraction in a nutshell*

- 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



Constraints

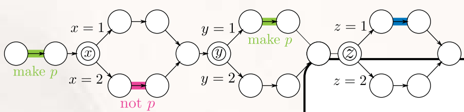
$\phi_{e_c} \Rightarrow \bigvee_i (a_i = e_{p_i} \wedge \phi_{e_{p_i}})$	One candidate causal link must hold
$[\epsilon, \infty] : \{a_{e_c,p} = e_{p_i}\} \wedge \phi_{e_{p_i}} \wedge \phi_{e_c}$ from e_{p_i} to e_c	Producers precede consumers
$\neg\phi_C$	Temporal conflicts (Conrad 2009)
$\neg\phi_C$	Threat resolutions
$[\epsilon, \infty] : \phi_C$ from e_c to e_{t_i}	Threat resolutions
$[\epsilon, \infty] : \phi_C$ from e_{p_i} to e_{t_i}	Threat resolutions
$[\epsilon, \infty] : \{b_{e_c,p,e_{p_i},e_{t_i}} = 1\} \wedge \phi_C$ from e_{t_i} to e_{p_i} $[\epsilon, \infty] : \{b_{e_c,p,e_{p_i},e_{t_i}} = 2\} \wedge \phi_C$ from e_c to e_{t_i}	Threat resolutions

$$\Downarrow$$

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

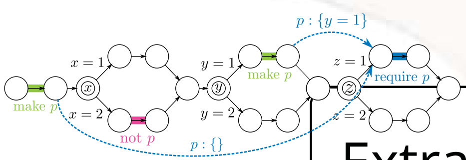
Offline

Online



Labeled APSP

$\Downarrow$

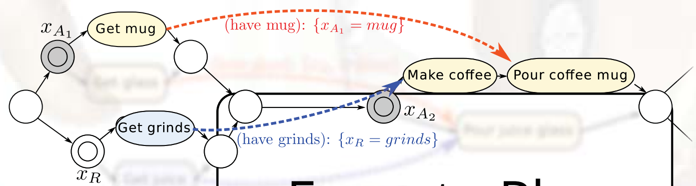


Extract Labeled Causal Links

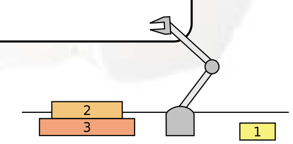
$\Downarrow$

$$\begin{cases} z=1 \Rightarrow (a_c=1 \lor (a_c=2 \land y=1)) \\ \neg(a_c=1 \land x=2 \land z=1) \end{cases}$$

Compile Constraints



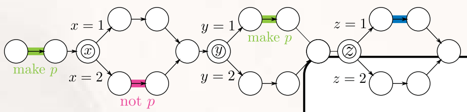
Execute Plan



Compile constraints with ATMS

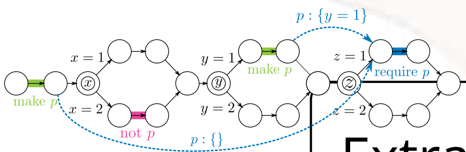
- Compile constraints for fast, online reactivity.
- Assumption-based Truth Maintenance System (ATMS): knowledge base permitting fast querying of assumptions
- Fast online queries without re-solving CSP:
    - “Can robot pick up coffee grounds now?”
    - “Is plan still feasible?”
- Internally, employs label propagation to pre-compute sets of consistent solutions

Offline



Labeled APSP

⇓



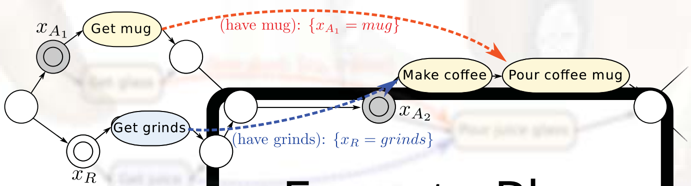
Extract Labeled Causal Links

⇓

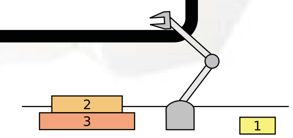
$$\begin{cases} z = 1 \Rightarrow (a_c = 1 \vee (a_c = 2 \wedge y = 1)) \\ \neg(a_c = 1 \wedge x = 2 \wedge z = 1) \end{cases}$$

Compile Constraints

Online

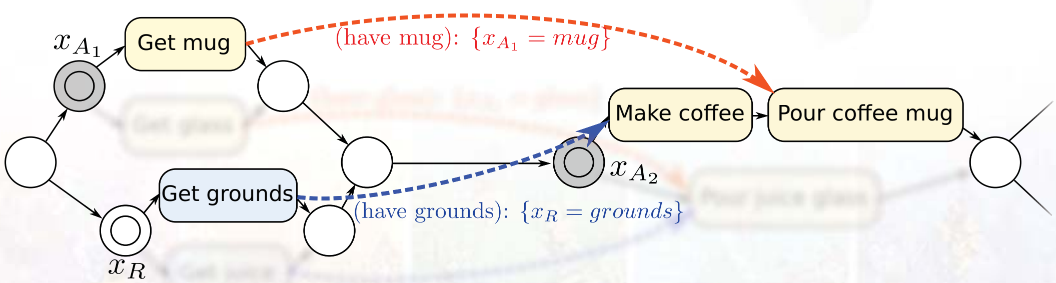


Execute Plan



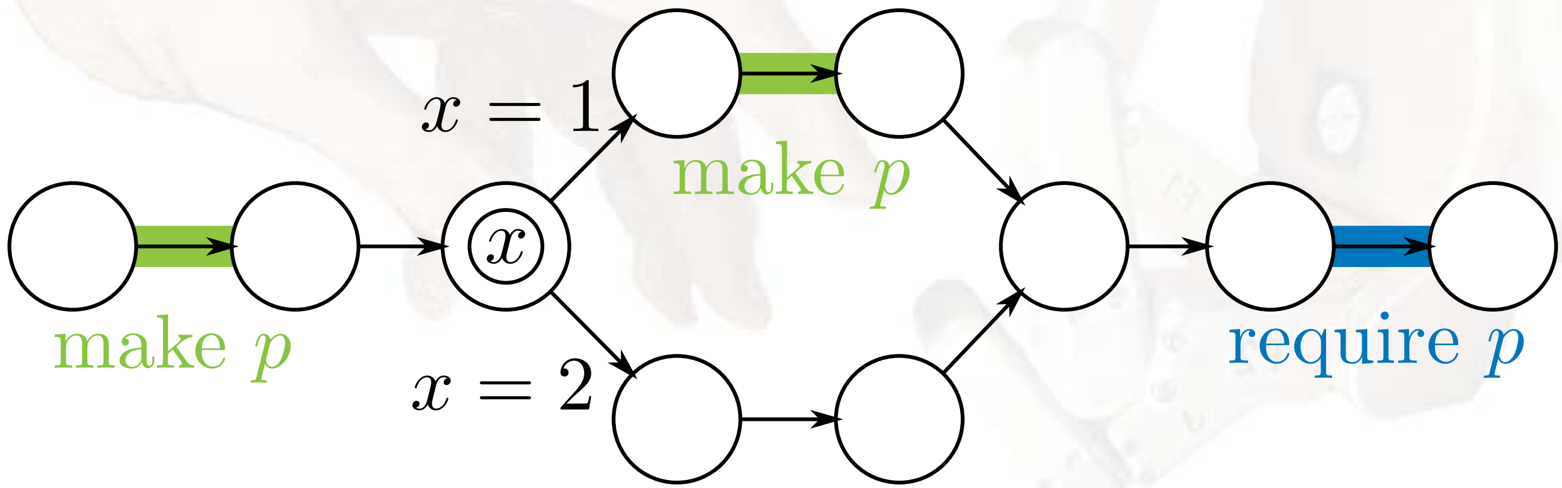
Online execution

- 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.



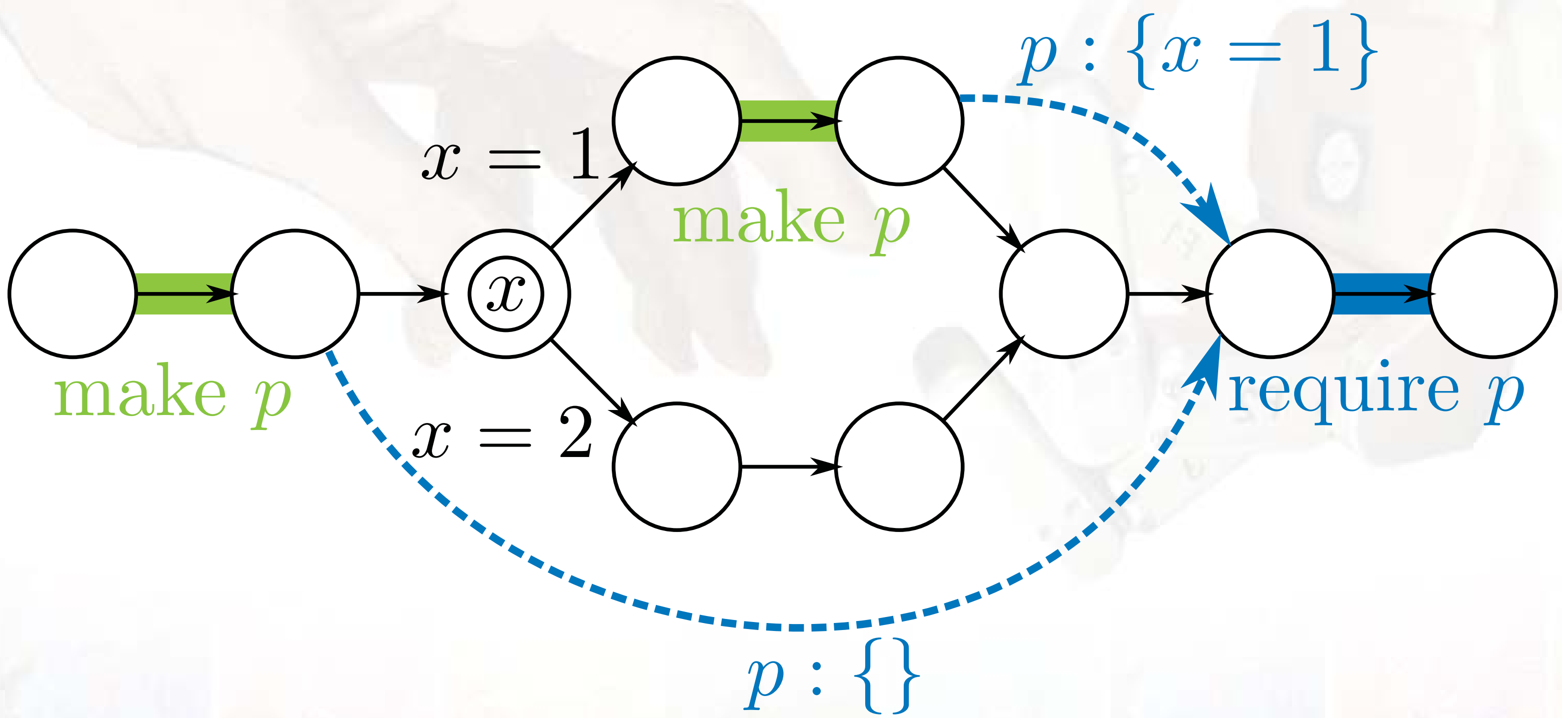
Online causal link execution monitoring

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



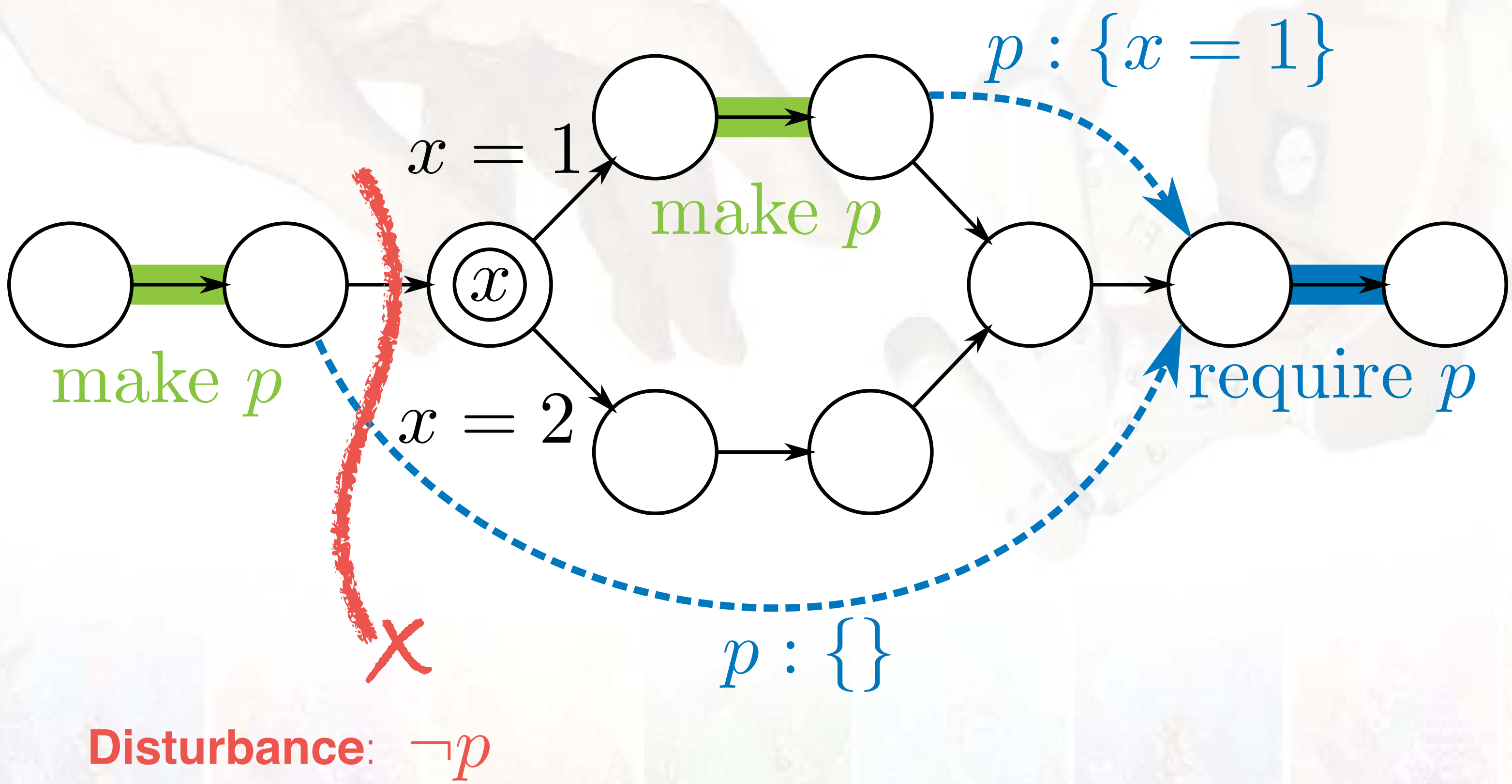
Online causal link execution monitoring

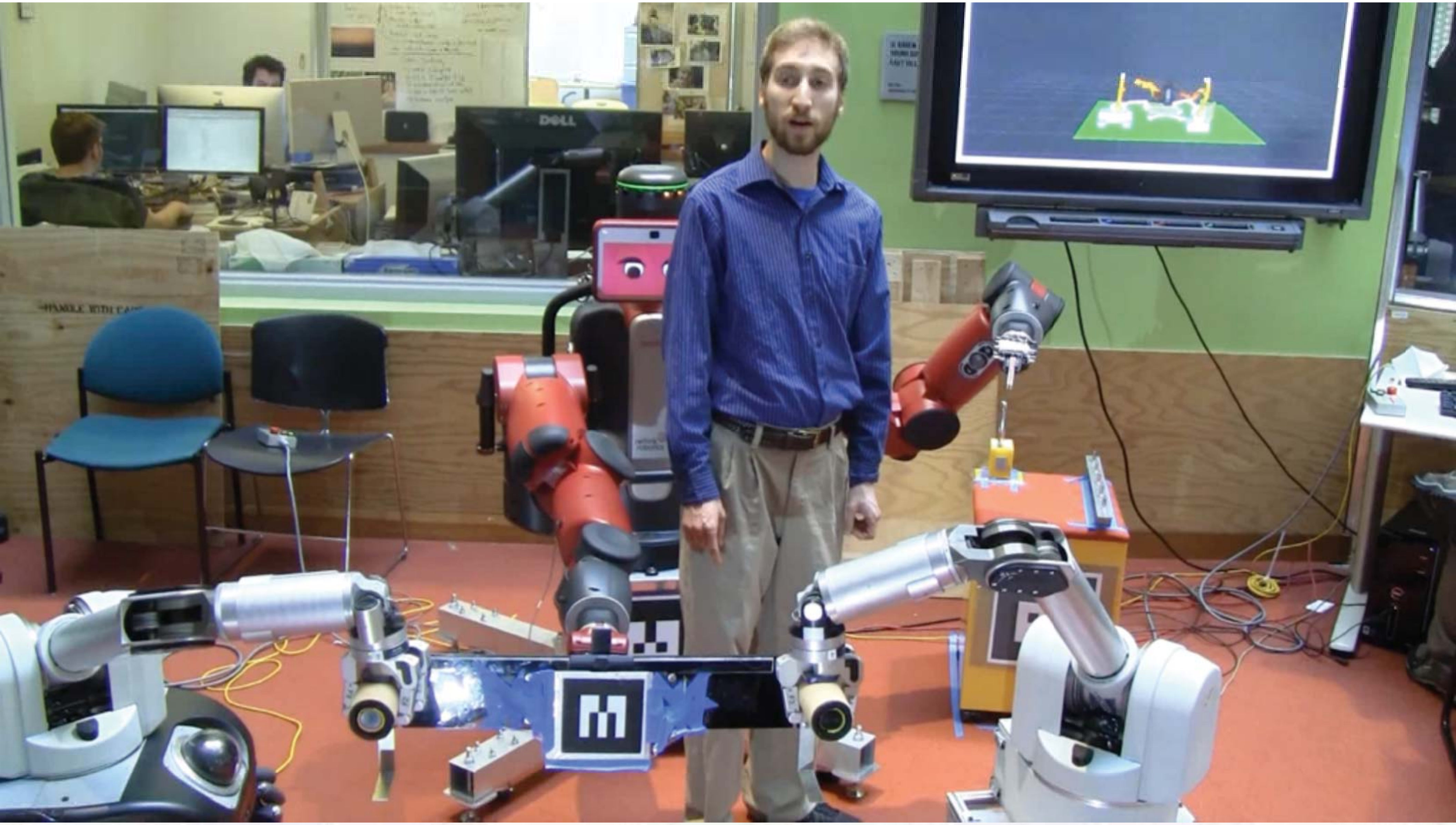
- Detects potential problems immediately
- Allows recovery actions (if modeled in plan)



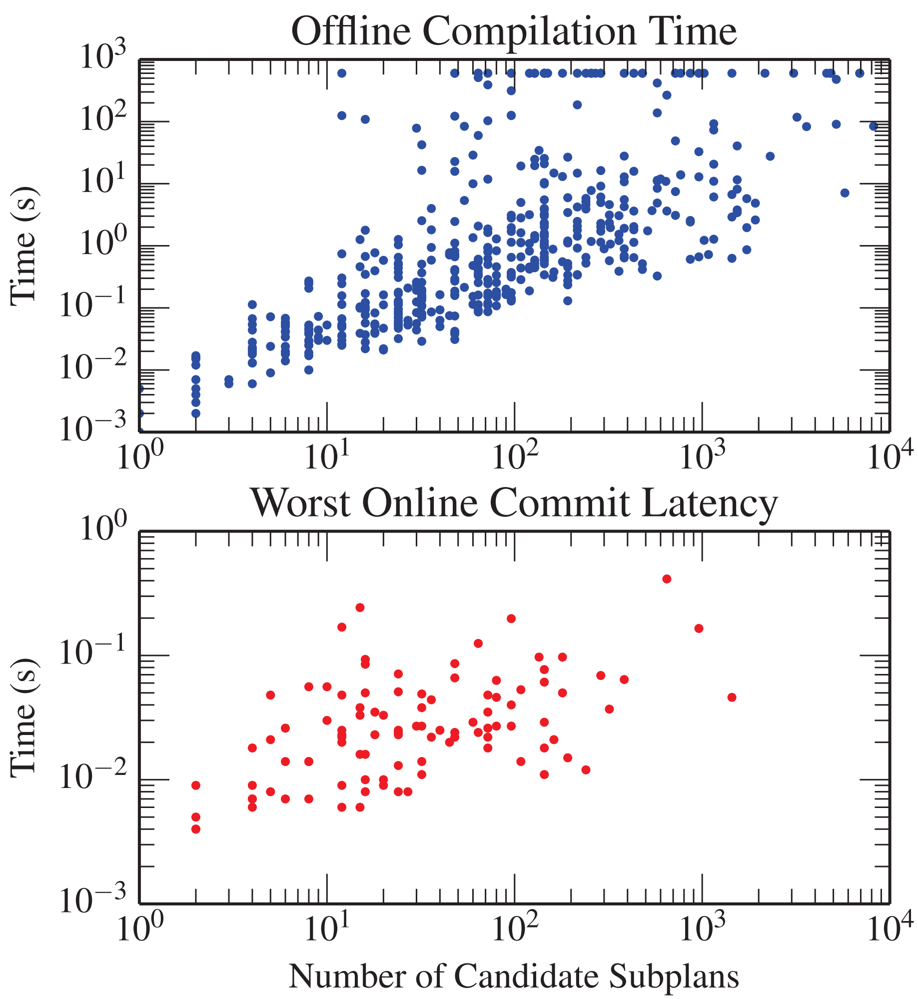
Online causal link execution monitoring

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




DELL

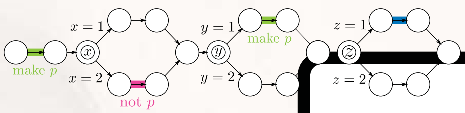
Experimental results



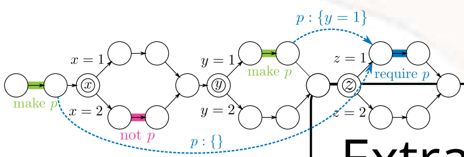
- Randomly-generated TPNU's with randomly-generated causal link structure (probably harder)
- Compilation time roughly proportional to candidate subplans, (large variance)
- Reactive online performance

Offline

Online



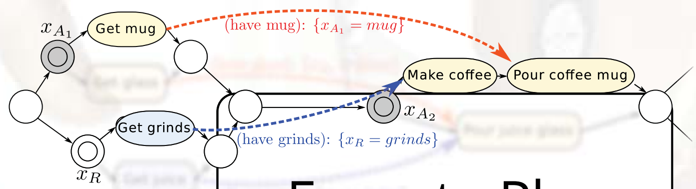
Labeled APSP



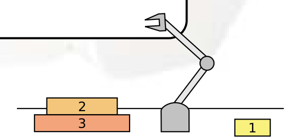
Extract Labeled Causal Links

$$\begin{cases} z=1 \Rightarrow (a_c=1 \lor (a_c=2 \land y=1)) \\ \neg(a_c=1 \land x=2 \land z=1) \end{cases}$$

Compile Constraints

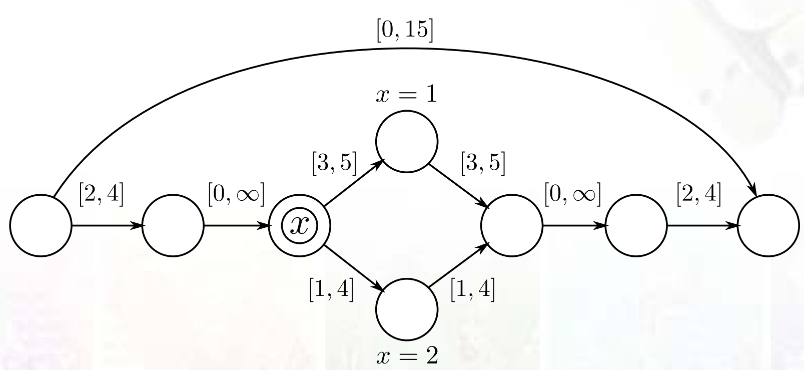


Execute Plan

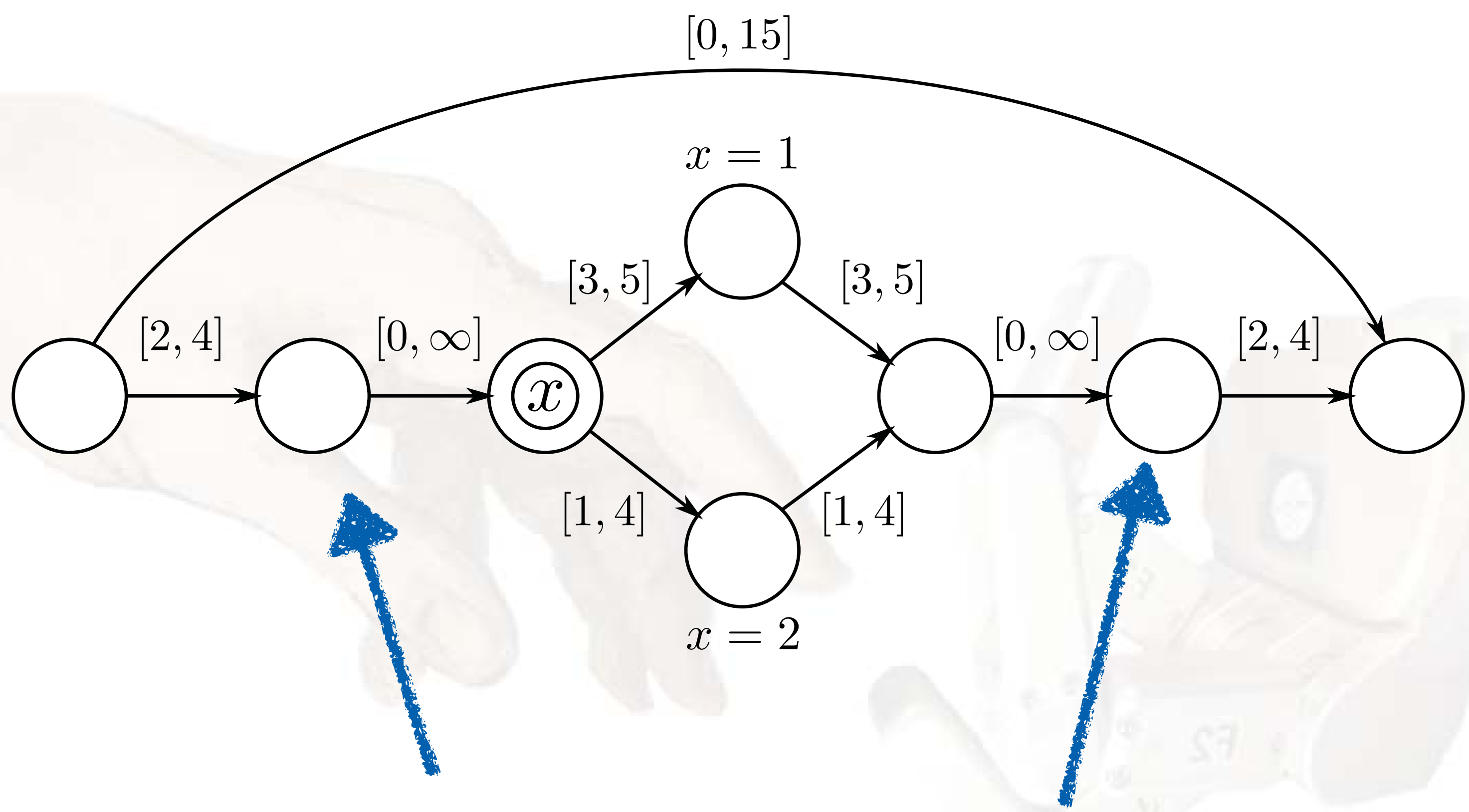


Labeled all-pairs shortest path (APSP)

- Developed in (Conrad 2009)
- Three purposes:
  1. Dispatchable form for online execution
  2. Ordering over events for causal link extraction
  3. Temporal conflict extraction

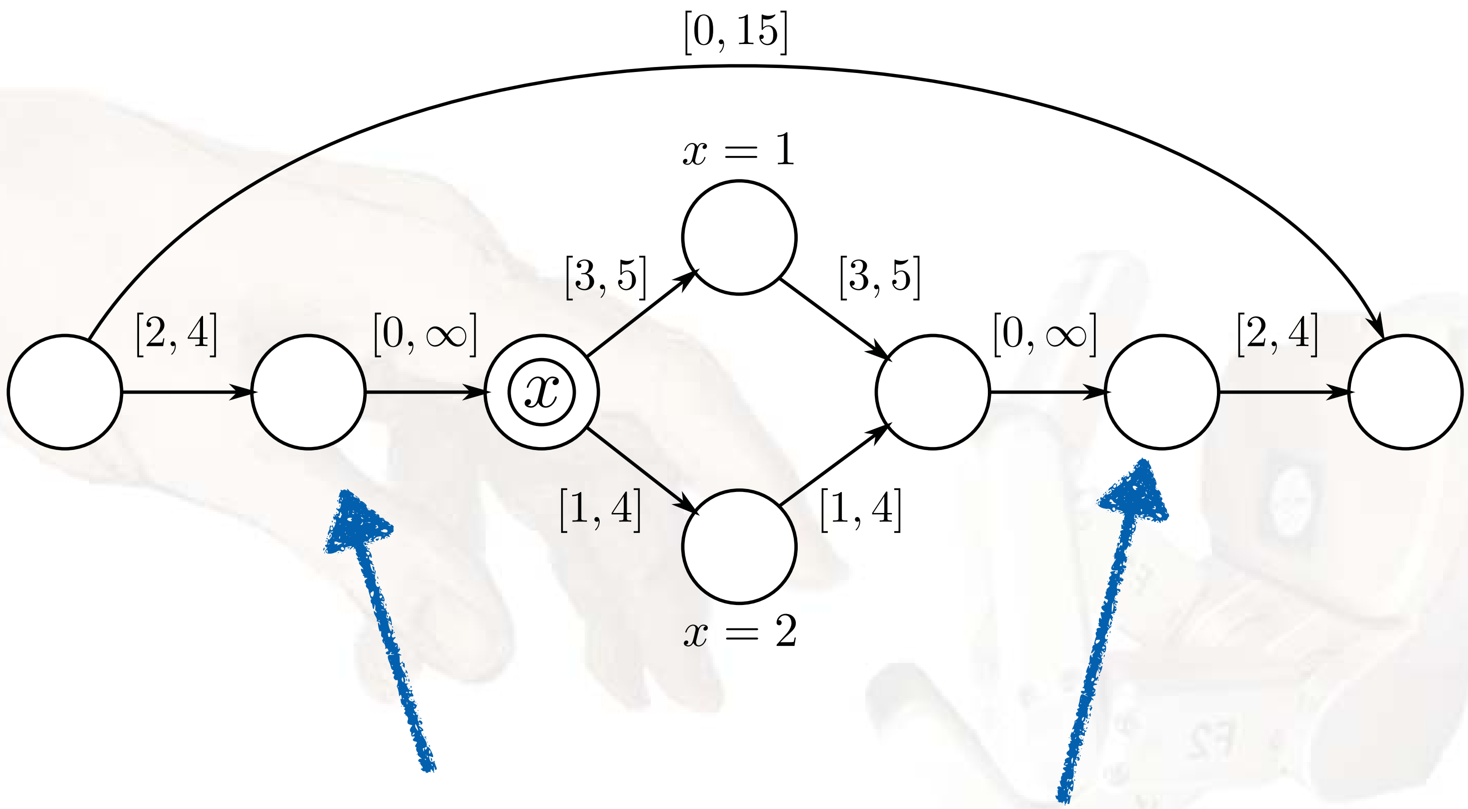


Labeled all-pairs shortest path (APSP)



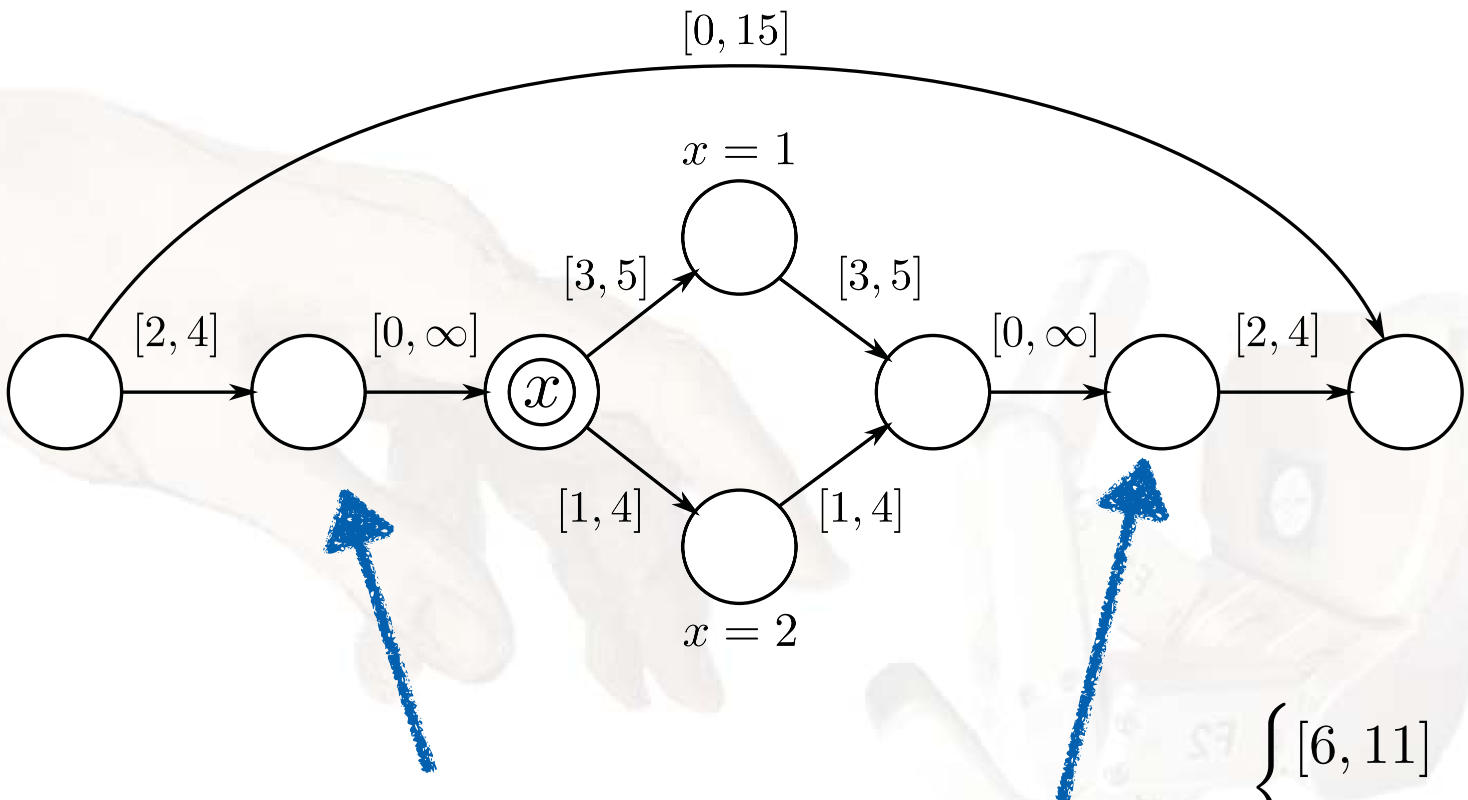
- What is temporal distance between these events?

Labeled all-pairs shortest path (APSP)



- What is temporal distance between these events?
  - Event dependent: $\begin{cases} [6, 11] & \text{if } x = 1 \\ [2, 11] & \text{if } x = 2 \end{cases}$

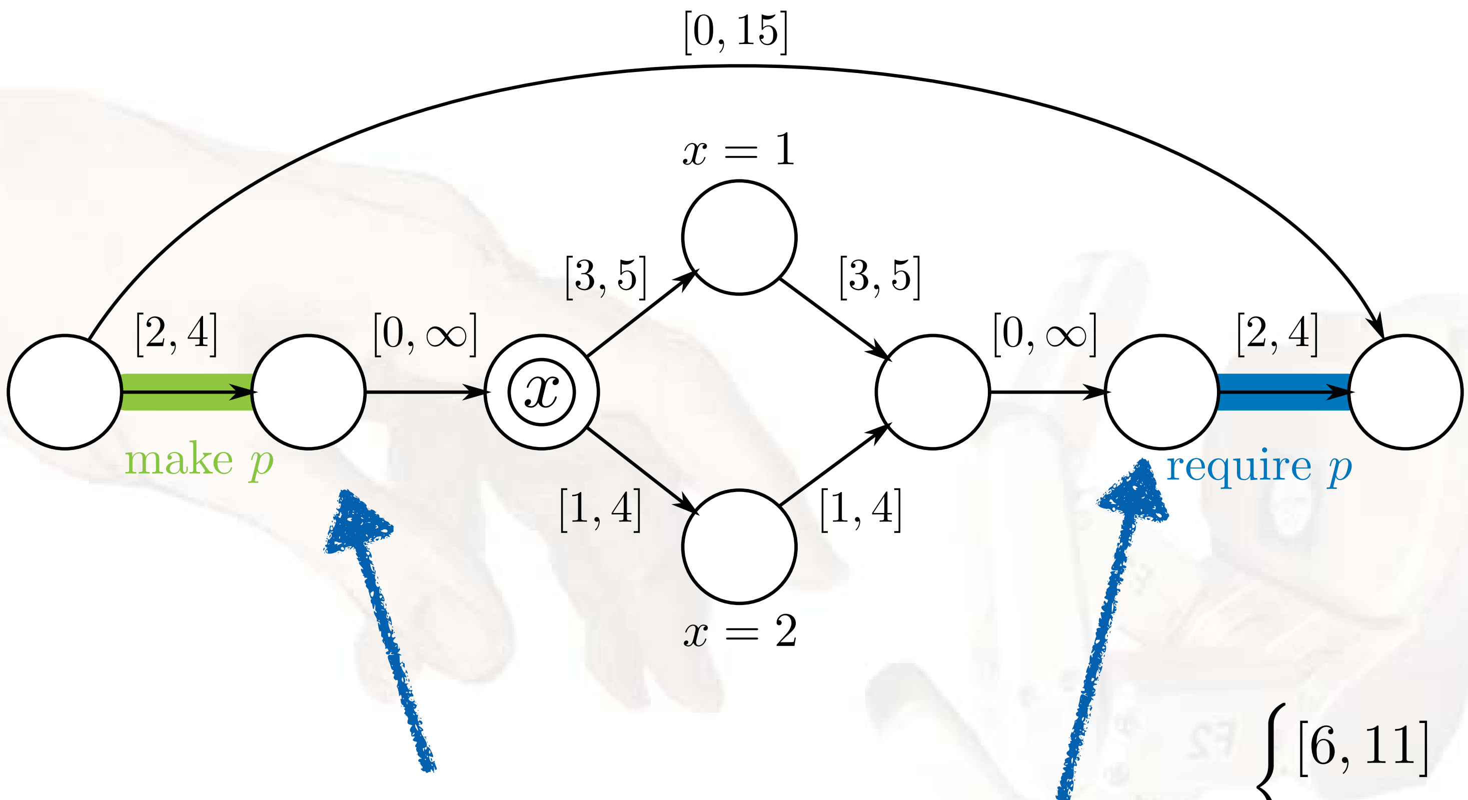
Labeled all-pairs shortest path (APSP)



$$\begin{cases} [6, 11] & \text{if } x = 1 \\ [2, 11] & \text{if } x = 2 \end{cases}$$

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

Labeled all-pairs shortest path (APSP)



$$\begin{cases} [6, 11] & \text{if } x = 1 \\ [2, 11] & \text{if } x = 2 \end{cases}$$

- 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, ϕ_i) dominates (a_j, ϕ_j) iff $a_i < a_j$ and ϕ_i subsumes ϕ_j

LVS introduced in (Conrad 2009)

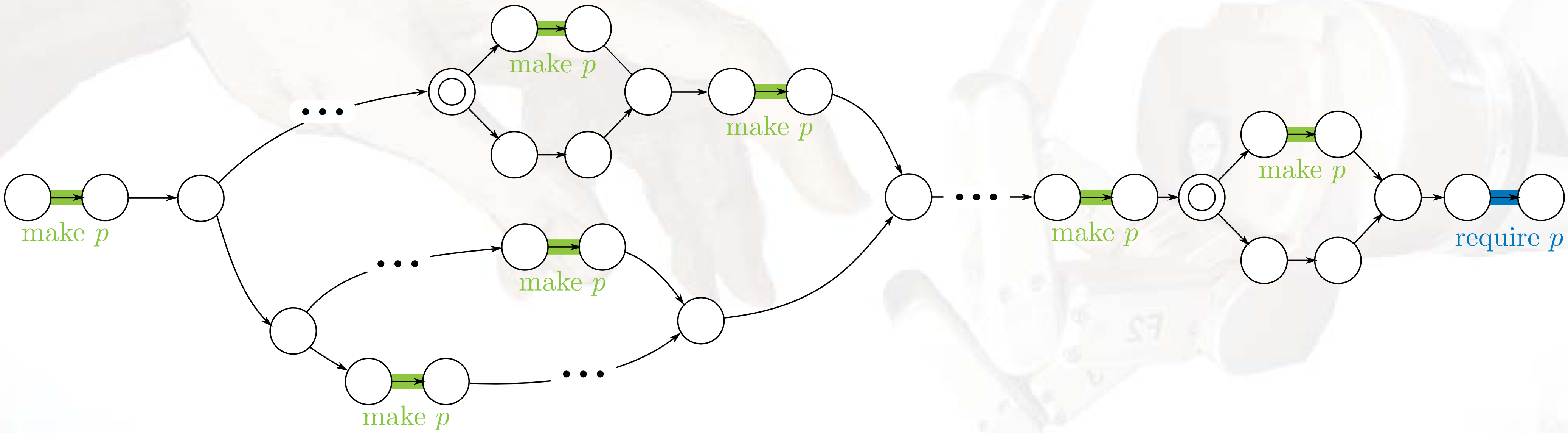
Operations on LVS's

- 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

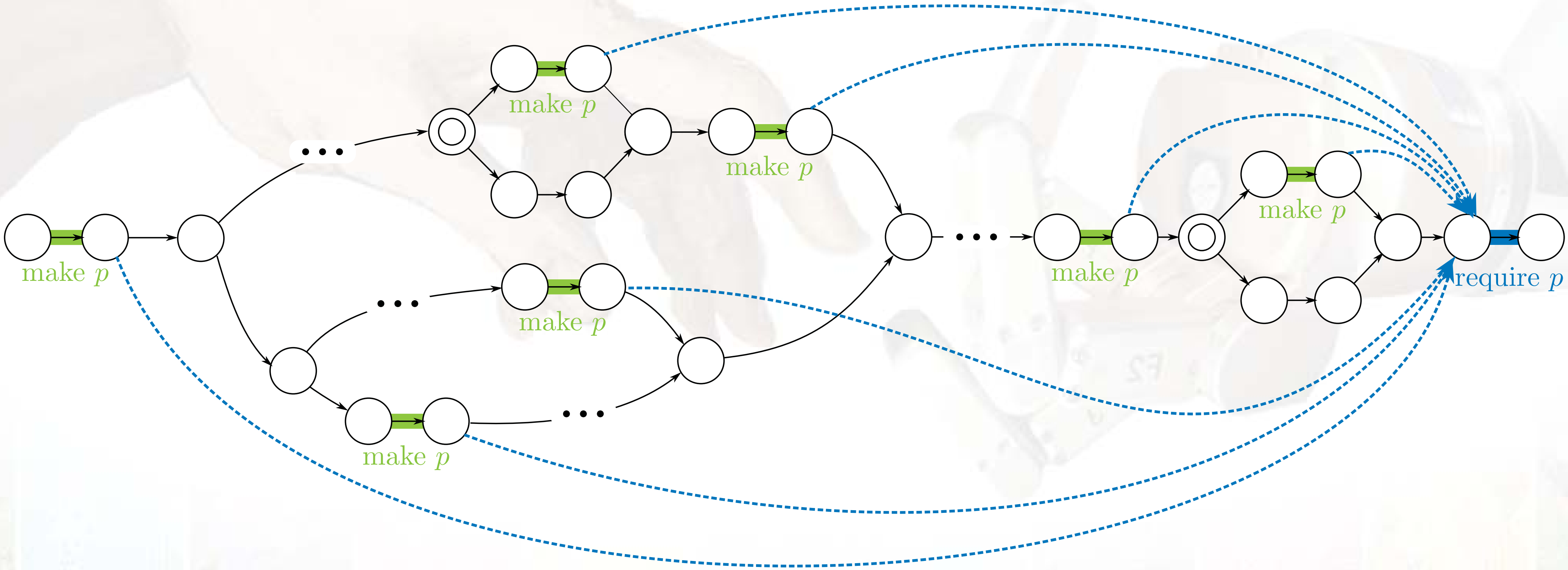
Labeled all-pairs shortest path (APSP)

- 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

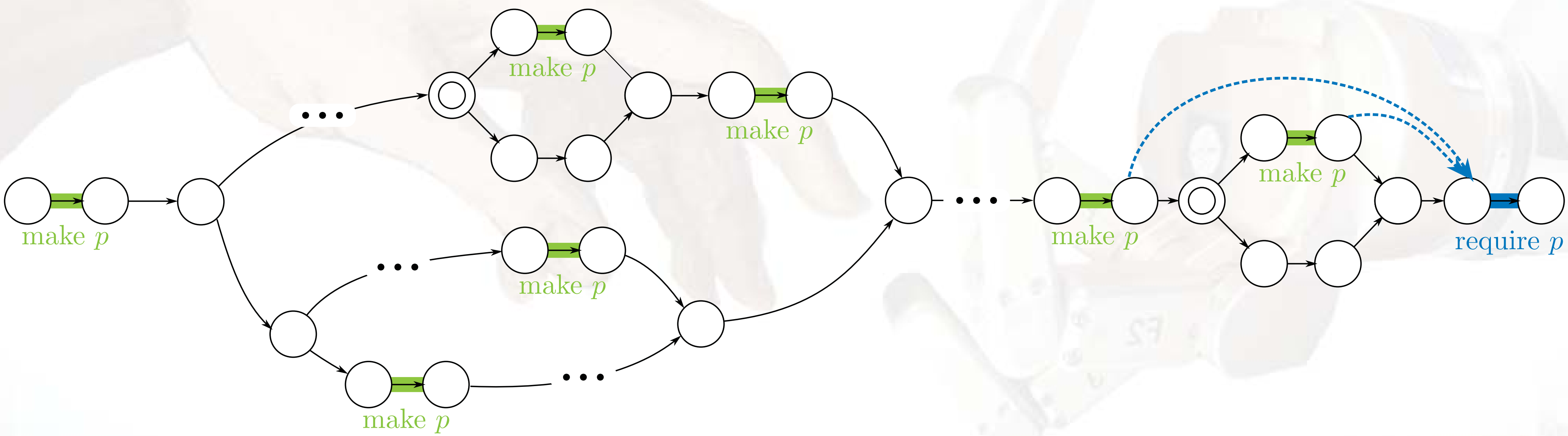
Optimization: Causal link dominance



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

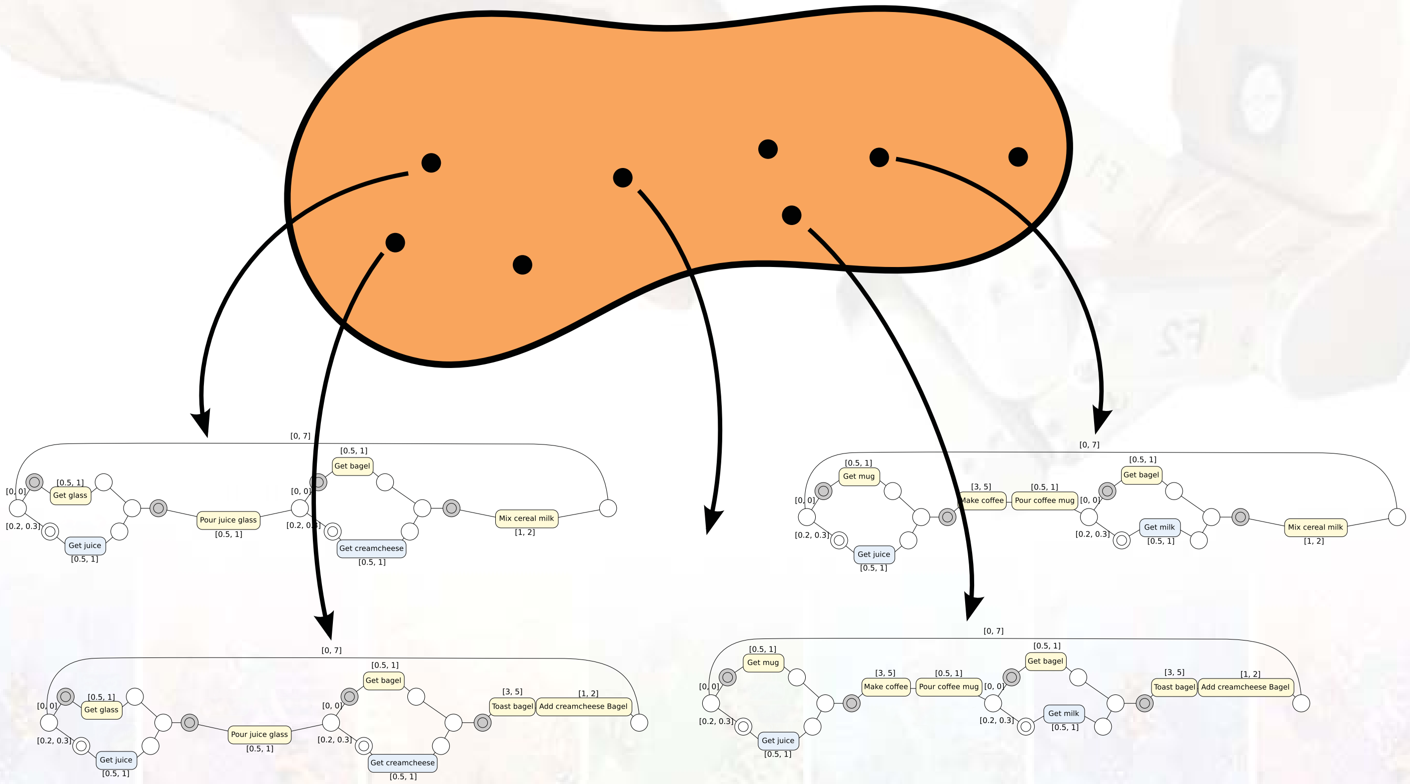
- Rank intent/adaptation hypotheses.
  - Probability: consider only likely human intents first
- Richer model for state & human
  - Hybrid state with continuous, spatial variables
  - Hybrid causal links via flow tubes / funnels
- Robot to actively influence human
  - Actively ask clarification questions
  - Informing human of increasingly likely failures (deadlines getting close, likely violated causal link, etc.)

Questions

Backup slides

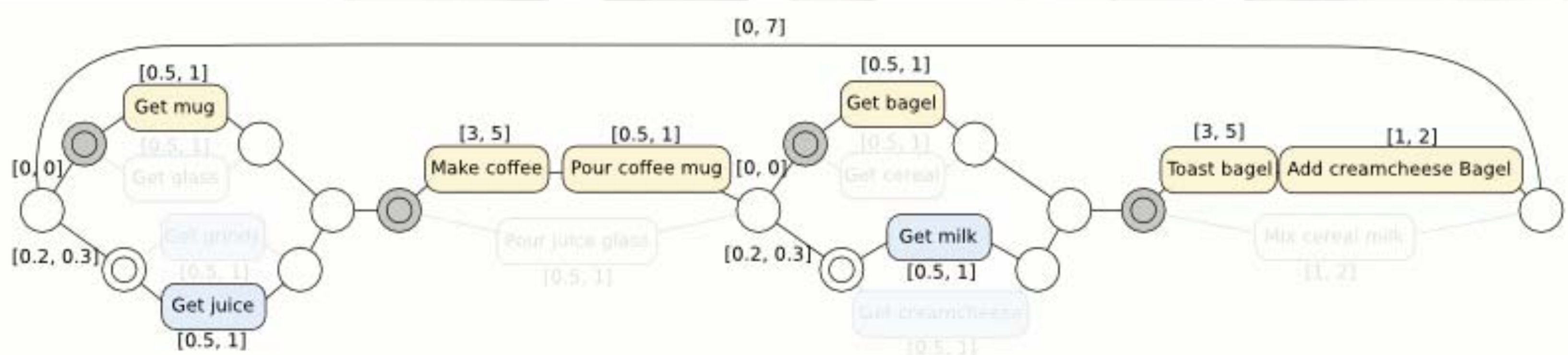
Environments represent sets of subplans

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



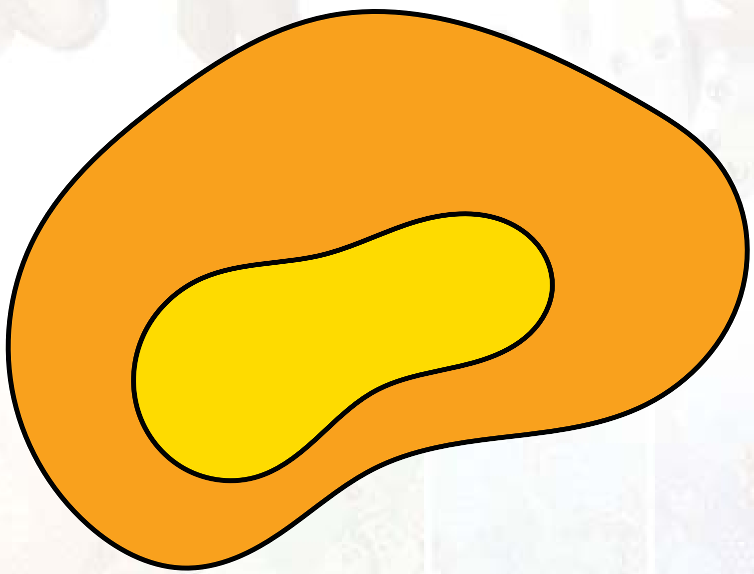
Environments represent sets of subplans

- Ex., $\{x_{R1} = juice, x_{A3} = bagel\}$ represents:

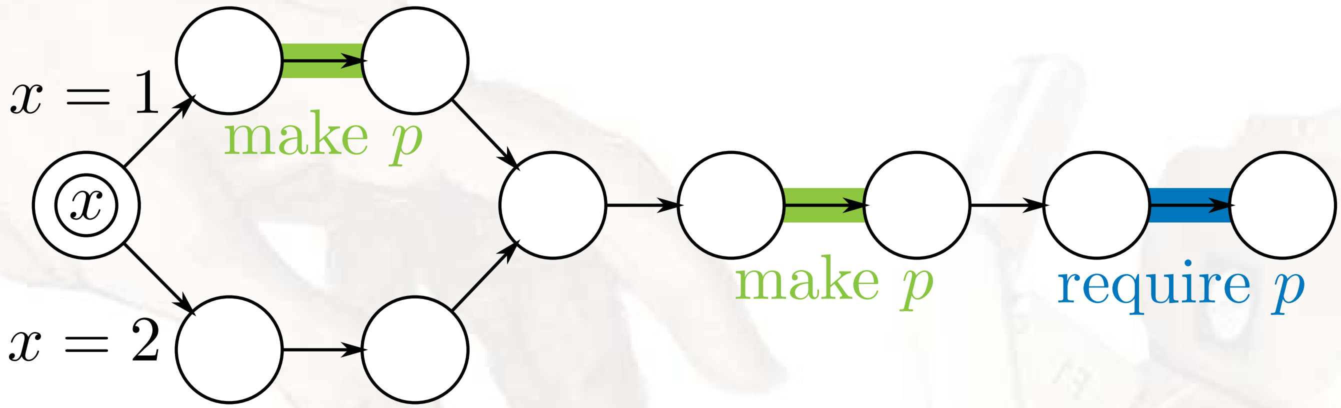


Environments and subsumption

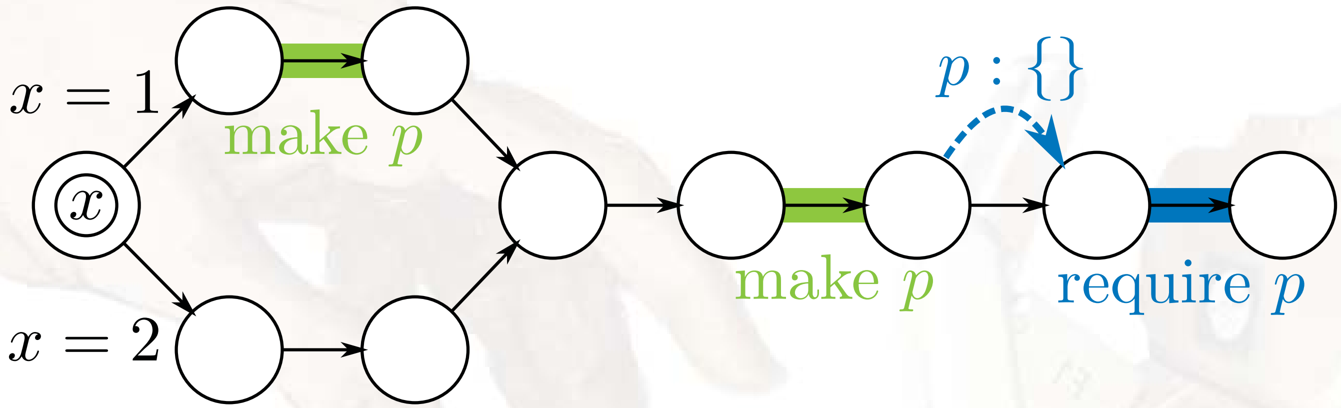
- Environment e_1 **subsumes** e_2 iff e_2 contains all assignments in e_1
  - ex., $\{x_{R1} = juice\}$ subsumes $\{x_{R1} = juice, x_{A3} = bagel\}$
- Intuitively, all subplans represented by e_2 also represented by e_1 (subset)



Labeled causal links

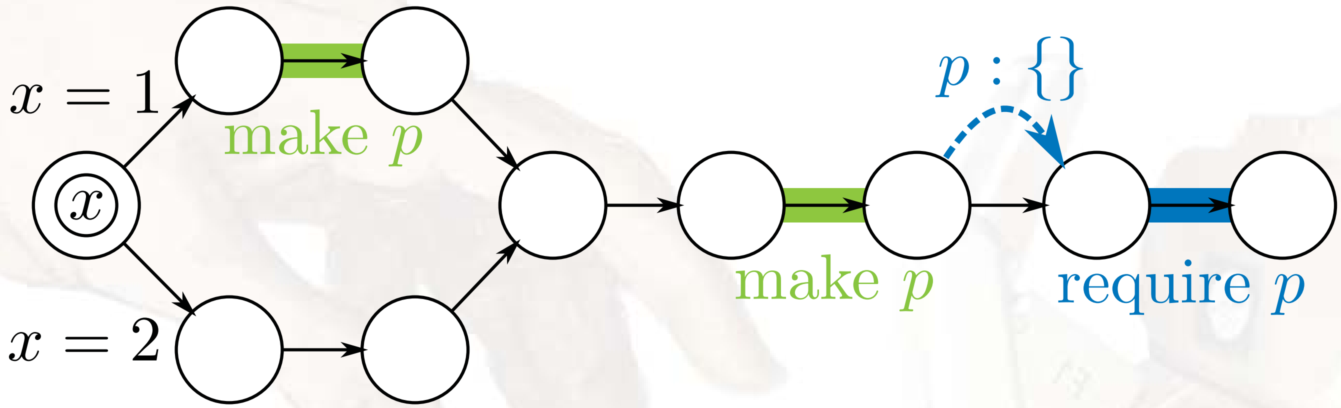


Labeled causal links



- 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.

Extraction algorithm

Algorithm 8: ExtractCausalLinkConstraints()

Input:
Output:

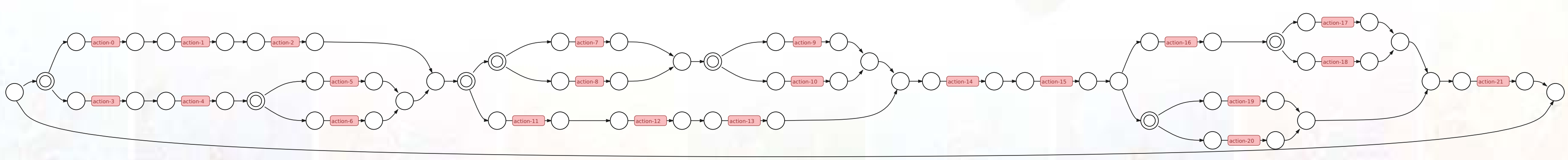
foreach $e_c \in \mathcal{E}$ **do**
  foreach $p \in$ Preconditions(e_c) **do**
    $\xi \leftarrow$ new LVS $\{\}$ with relation $\prec$
    foreach $e \neq e_c \in \mathcal{E}$ *with* p *or* $\neg p$ *in* Effects(e) **do**
      if *not* $e_c \prec e$ *and* $\phi_e \wedge \phi_{e_c}$ *is feasible(!)* **then**
        AddLVS$((e, \phi_e), \xi)$
      end
    end
    Separate ξ into P and T
    Create decision variable $a_{e_c,p}$ with domain P
    AddConstraint$\left(\phi_{e_c} \Rightarrow \bigvee_i (a_i = e_{p_i} \wedge \phi_{e_{p_i}})\right)$
    foreach $e_{p_i} \in P$ *where not* $e_{p_i} \prec e_c$ **do**
      Add $[\epsilon, \infty] : \{a_{e_c,p} = e_{p_i}\} \wedge \phi_{e_{p_i}} \wedge \phi_{e_c}$ from e_{p_i} to e_c
    end
    foreach $e_{p_i} \in P$ **do**
      foreach $e_{t_i} \in T$ **do**
        $\phi_C \leftarrow \{a_{e_c,p} = e_{p_i}\} \wedge \phi_{e_{p_i}} \wedge \phi_{e_{t_i}} \wedge \phi_{e_c}$
        if $e_{p_i} \prec e_{t_i}\big|_{\phi_C}$ *and* $e_{t_i} \prec e_c\big|_{\phi_C}$ **then**
          AddConstraint$\left(\neg\phi_C\right)$
        else if $e_{p_i} \prec e_{t_i}\big|_{\phi_C}$ *and* $e_{t_i} \parallel e_c\big|_{\phi_C}$ **then**
          Add $[\epsilon, \infty] : \phi_C$ from e_c to e_{t_i}
        else if $e_{p_i} \parallel e_{t_i}\big|_{\phi_C}$ *and* $e_{t_i} \prec e_c\big|_{\phi_C}$ **then**
          Add $[\epsilon, \infty] : \phi_C$ from e_{p_i} to e_{t_i}
        else if $e_{p_i} \parallel e_{t_i}\big|_{\phi_C}$ *and* $e_c \parallel e_{t_i}\big|_{\phi_C}$ **then**
          Create decision variable $b_{e_c,p,e_{p_i},e_{t_i}}$ with domain $\{1, 2\}$
          Add $[\epsilon, \infty] : \{b_{e_c,p,e_{p_i},e_{t_i}} = 1\} \wedge \phi_C$ from e_{t_i} to e_{p_i}
          Add $[\epsilon, \infty] : \{b_{e_c,p,e_{p_i},e_{t_i}} = 2\} \wedge \phi_C$ from e_c to e_{t_i}
        end
      end
    end
  end
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



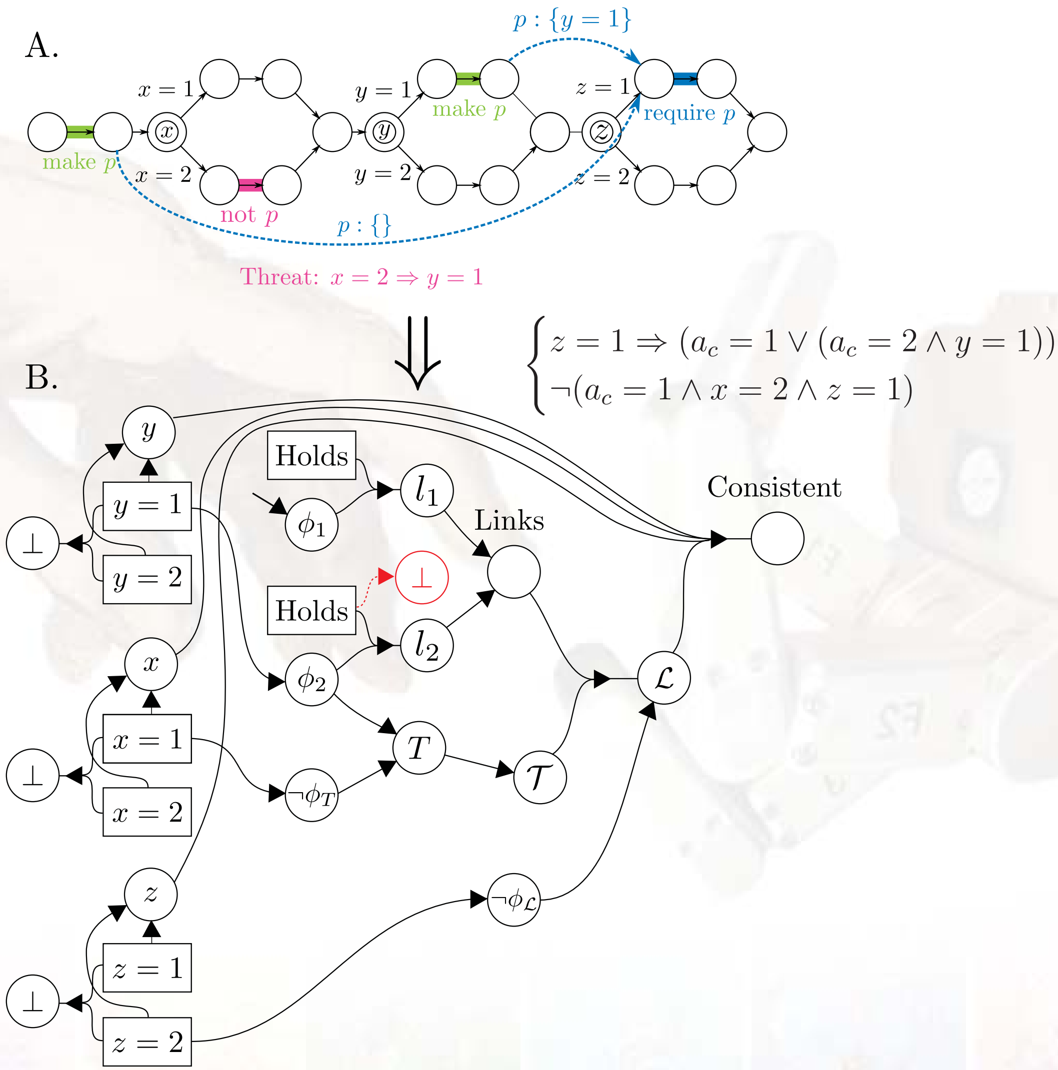
Labeled causal link extraction

- Uses an LVS, except:
  - Values are TPNU *events*, rather than *numbers*
  - Relation *R* is not < but rather succession (via labeled APSP):

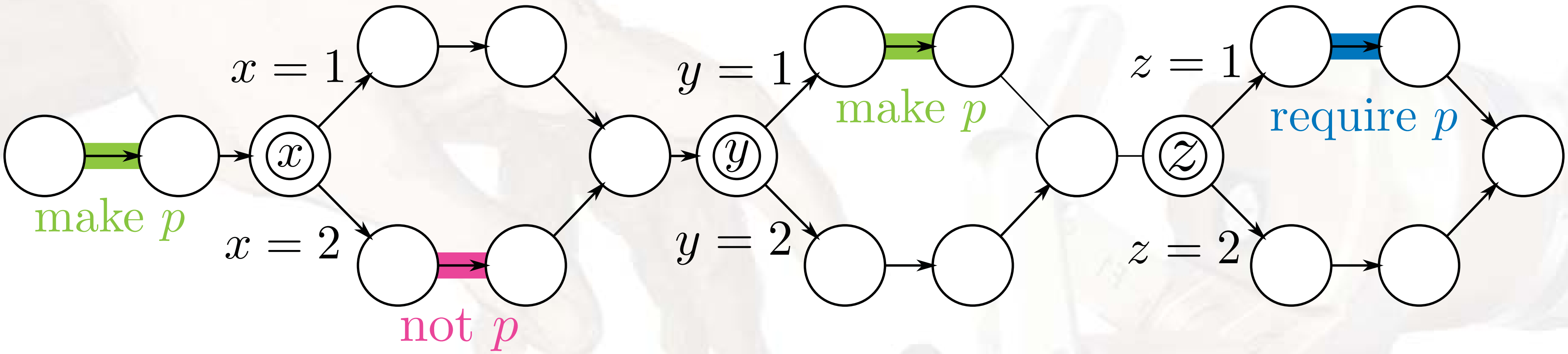
$$e_a \; R \; e_b = \begin{cases} \textsc{True} & \text{if } Q_{d_{e_a \to e_b}}(\phi_a \cup \phi_b) \leq 0 \\ \textsc{False} & \text{otherwise} \end{cases}$$

- 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)

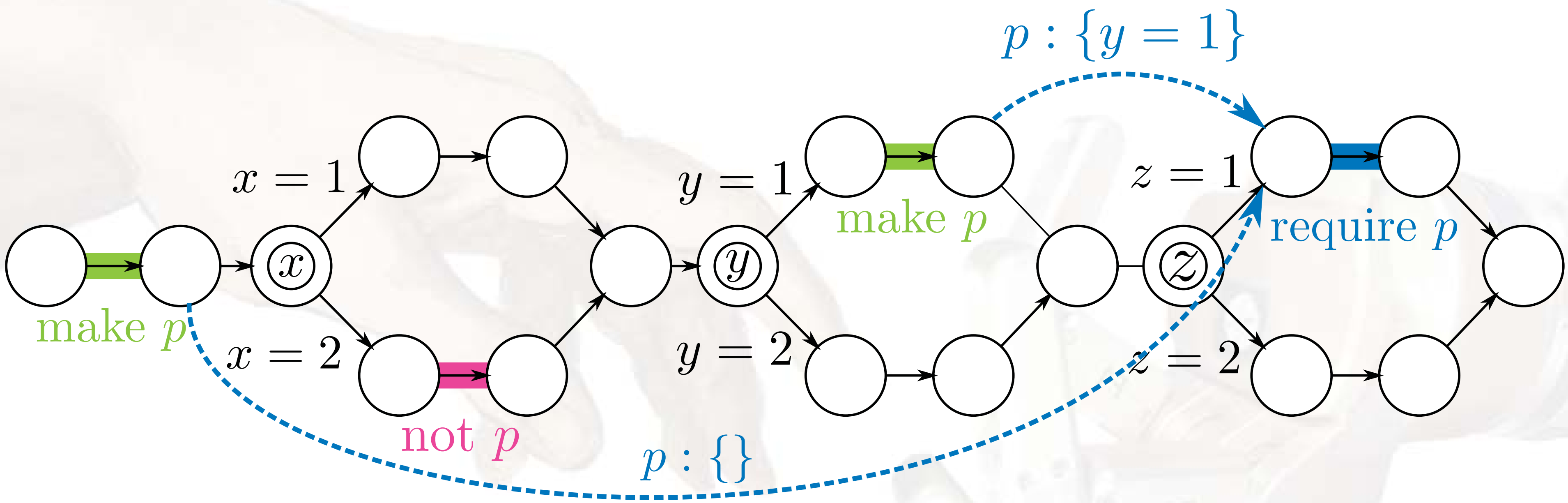
Encoding in an ATMS



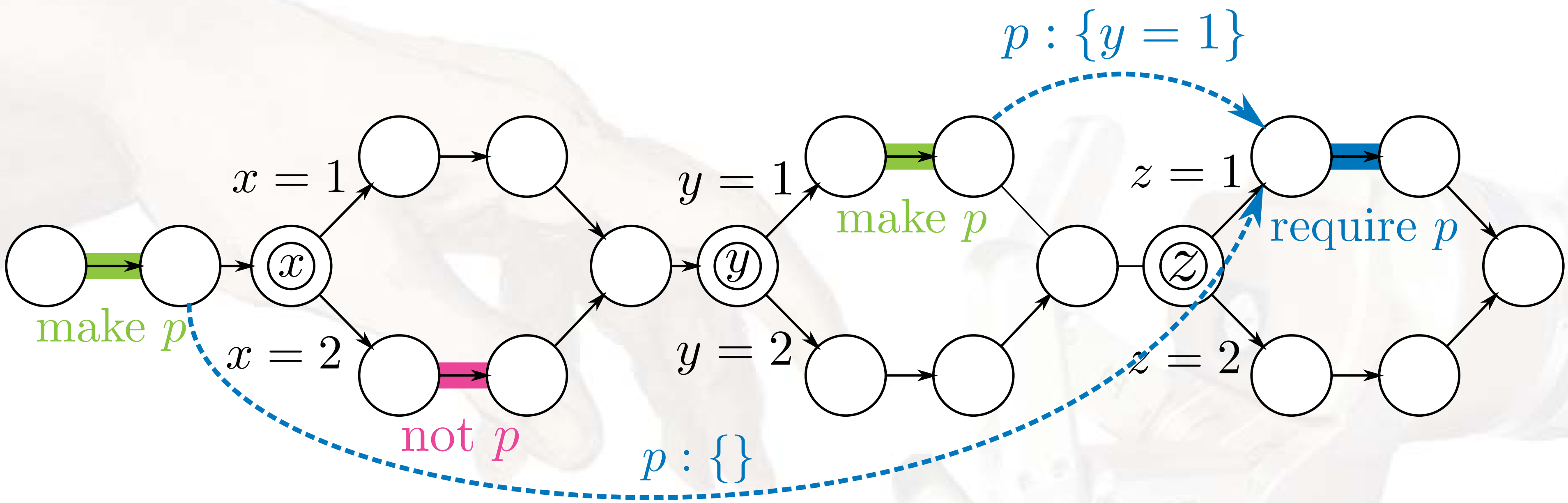
A basic threat



A basic threat

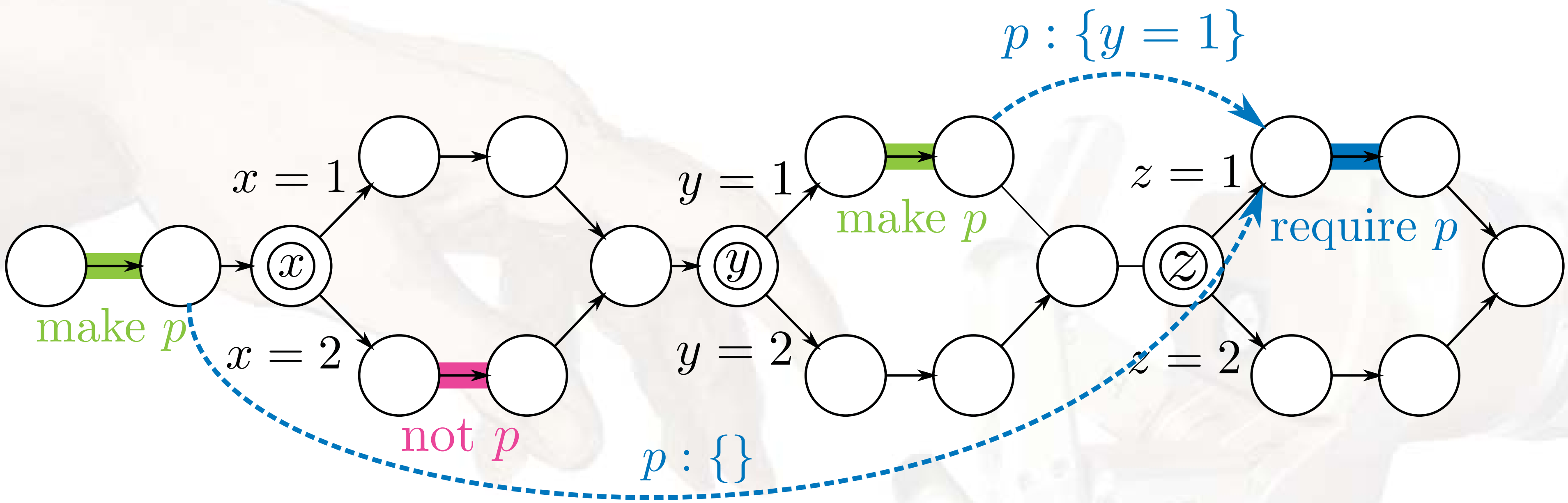


A basic threat



$$a_c = \begin{cases} 1 & \text{if first causal link enforced} \\ 2 & \text{if second causal link enforced} \end{cases}$$

Extracting propositional constraints



$$a_c = \begin{cases} 1 & \text{if first causal link enforced} \\ 2 & \text{if second causal link enforced} \end{cases}$$

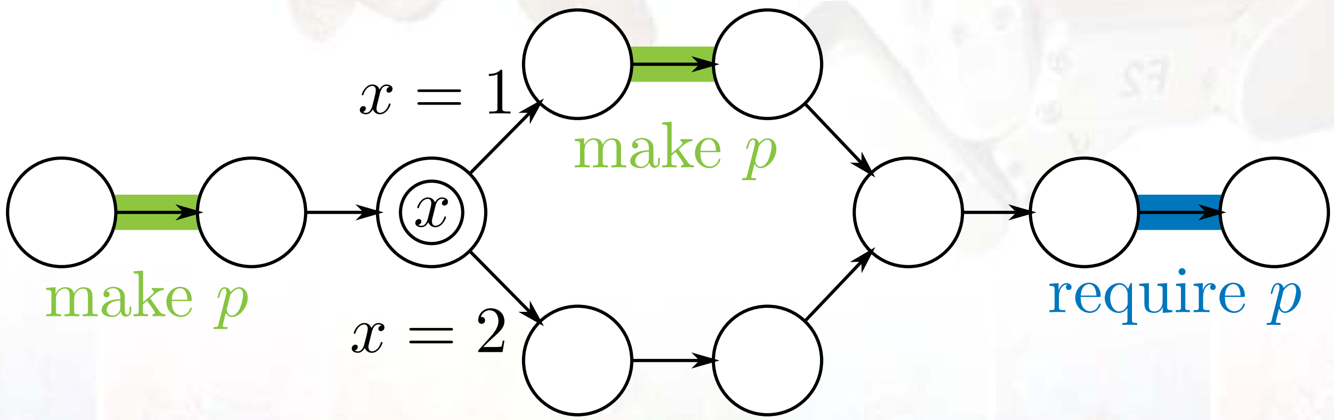
$$\begin{cases} z = 1 \Rightarrow (a_c = 1 \vee (a_c = 2 \wedge y = 1)) \\ \neg(a_c = 1 \wedge x = 2 \wedge z = 1) \end{cases}$$

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