

# Applications of Automated Planning for Business Process Management

Andrea Marrella, Tathagata Chakraborti

marrella@diag.uniroma1.it

Tathagata.Chakraborti1@ibm.com

**Tutorial @ BPM 2021**

September 7, 2021



**SAPIENZA**  
UNIVERSITÀ DI ROMA





1. New trends in Business Process Management
2. Basics of Automated Planning
3. Automated Planning along the BPM Life-cycle
  - Declarative modeling
  - Trace alignment against declarative process models
  - Interpretability
4. Conclusions



- 1. New trends in Business Process Management**
2. Basics of Automated Planning
3. Automated Planning along the BPM Life-cycle
  - Declarative modeling
  - Trace alignment against declarative process models
  - Interpretability
4. Conclusions



- **Traditional BPM techniques** focus on supporting well structured business processes in highly controlled domains.
- **New trend:** BPM research is expanding towards new challenging domains (healthcare, smart manufacturing, etc.) characterized by:
  - ever-changing requirements;
  - unpredictable and cyber-physical environments;
  - increasing amounts of data that influence the running processes.
- BPM systems need techniques that go **beyond hard-coded solutions** and are capable of **autonomous behavior**.
- The challenge of building physical devices that act autonomously is **at the center of the AI research** from its origins.



- The rise of **AI-based** techniques for BPM:

- Keynote of **Rick Hull** at **BPM 2016**: “*Rethinking BPM in a Cognitive World: Transforming How We Learn and Perform Business Processes*”
- Keynote of **Miguel Valdés** at **BPM 2017**: “*Intelligent continuous improvement, when BPM meets AI*”
- Keynote of **Rama Akkiraju** at **BPM 2020**: “*How to infuse AI into Business Processes? A practitioner's perspective on best practices*”
- Keynote of **Giuseppe De Giacomo** at **BPM 2021**: “*Artificial Intelligence-based Declarative Process Synthesis for BPM*”
- **AI4BPM** workshop since 2017 organized within the BPM conference



- At the center of the problem of autonomous behavior is the **control problem** (or **action selection problem**).
  - *specify a controller that selects the action to do next*
- Traditional hard-coded solutions specify a **pre-scribed controller** in a high-level language.
  - ✓ They do not suffer combinatorial explosion.
  - The burden is all put on the programmer.
  - Hard-coded solutions are usually problem-dependent and tend to constraint the search in some way.
- The question of action selection for AI researchers is:
  - *What is the best way to intelligently constrain this search?*



- **Model-based approaches** to tackle autonomous behavior:
  - The controller is **derived automatically** from a model of the domain of interest, the actions, the current state, and the goal.
    - ✓ The models are all conceived to be general.
    - The problem of solving a model is computationally intractable.

In this tutorial, we introduce the basic ingredients of **automated planning** and the **PDDL language** for representing planning problems.



1. New trends in Business Process Management
2. **Basics of Automated Planning**
3. Automated Planning along the BPM Life-cycle
  - Declarative modeling
  - Trace alignment against declarative process models
  - Interpretability
4. Conclusions

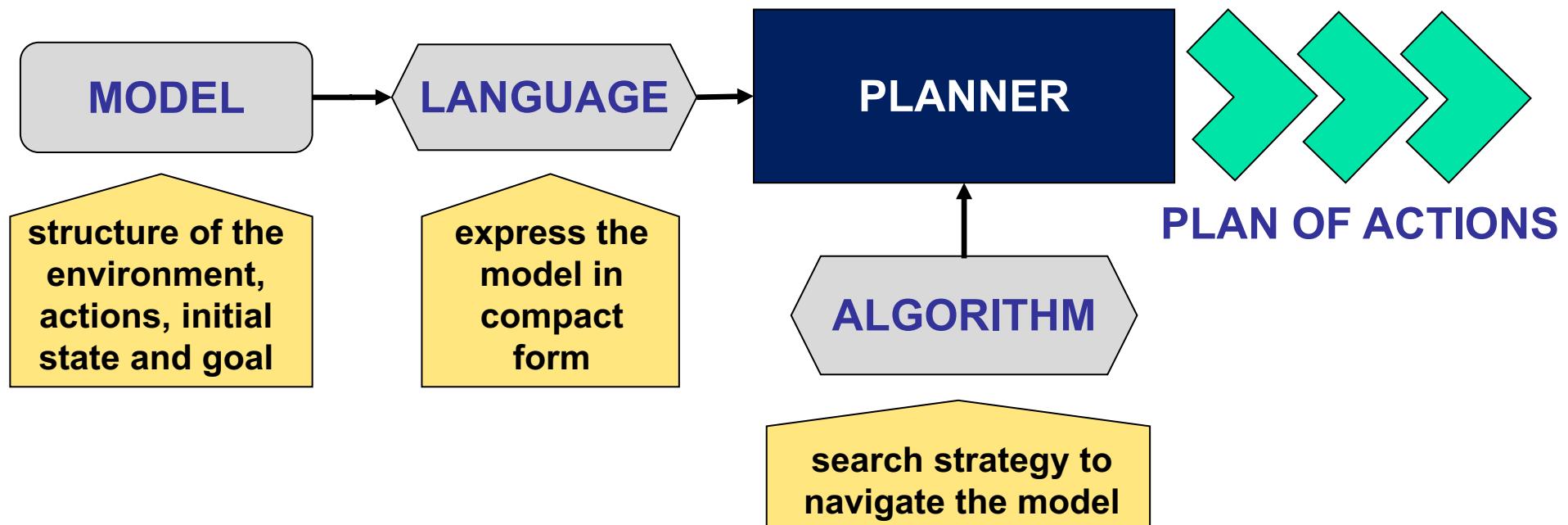


IBM

# Automated Planning

- In AI, **automated planning** is conceived as the:

***model-based approach for the automated synthesis of plans of actions to achieve goals.***





- **Several classes of planning models**, which depend on the properties of the problems to be represented:
  - full or partial observability of the current state;
  - uncertainty in the initial state (fully or partially known);
  - uncertainty in the actions dynamics (deterministic or not);
  - uncertainty represented by sets of states or probability distributions;
  - the type of feedback (full, partial or no state feedback).

Planning is computationally  
intractable even for the  
simplest models...

..BUT..

...classical planners  
solve efficiently  
real problems  
with hundreds  
of propositions!



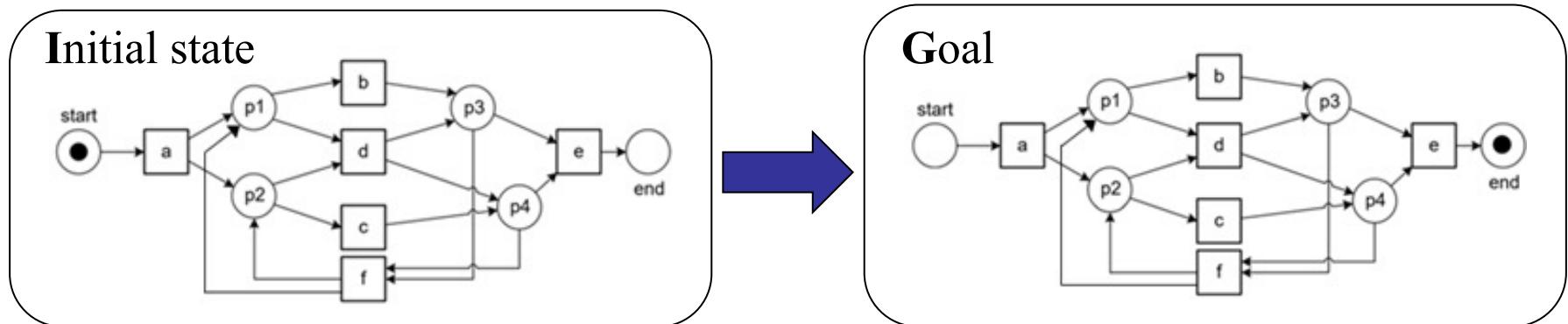
# Classical Planning Model

- **finite and discrete state space  $S$**
- **a known initial state  $I \in S$**
- **a set  $S_G \subseteq S$  of goal states**
- **actions  $A(s) \subseteq A$  applicable in each  $s \in S$**
- **a deterministic transition function  $s' = f(a, s)$  for  $a \in A(s)$**
- **positive action costs  $c(a, s)$**
- ❖ A **solution** or **plan** is a sequence of applicable actions  $\pi = a_0, \dots, a_n$  that maps  $I$  into  $S_G$ 
  - There are states  $s_0, \dots, s_{n+1}$  such that  $s_{i+1} = f(a_i, s_i)$  and  $a_i \in A(s_i)$  for  $i = 0, \dots, n$  and  $s_{n+1} \in S_G$
- ❖ A plan is **optimal** if it minimizes the sum of action costs  $\sum_{i=0, \dots, n} c(a_i, s_i)$ . If costs are all 1, plan cost is plan length.



# Example: Reachability in Petri Nets

- Given a Petri Net (PN) with an initial marking  $m_0$  and a target marking  $m_n$ , if there exists a sequence of transition firings  $\langle t_1 \dots t_n \rangle$  that leads from  $m_0$  to  $m_n$ , then  $m_n$  is said to be **reachable** from  $m_0$



- One available action: **firing**
  - If  $t$  is *enabled*, firing  $t$  changes the marking of the PN.
    - Each token from the input places of  $t$  is consumed, and one token is produced in any output place of  $t$ .



- The standard representation language for automated planners is known as the **Planning Domain Definition Language (PDDL)**.
- Components of a PDDL planning task:
  - **Objects**: Things in the world that interest us.
  - **Predicates**: Properties of objects that we are interested in; they can be true or false.
  - **Functions**: Variables that apply to zero or more objects and are assigned with a numeric value.
  - **Initial state**: The state of the world that we start in.
  - **Goal specification**: Things that we want to be true.
  - **Actions/Operators**: Ways of changing the state of the world.



- Problems in PDDL are expressed in two separate parts:
  - **PDDL Planning Domain PD** (available actions and predicates representing explicit representation of the world).
  - **PDDL Planning Problem PR** (objects, initial state **I** and goal condition **G**).
- A planner that takes in input a problem encoded in PDDL is said to be **domain-independent**, since it is able to automatically produce a plan without knowing what the actions and domain stand for.
- PDDL provides the ground for performing a **direct comparison** between different planning techniques and algorithms and evaluating against classes of problems.



- Domain files look like this:

```
(define (domain <domain name>)
  <PDDL code for predicates and functions>
  <PDDL code for first action>
  [...]
  <PDDL code for last action>
)
```

- <domain name> is a string that identifies the planning domain, e.g., petri-net.



- Problem files look like this:

```
(define (problem <problem name>)
  (:domain <domain name>)
  <PDDL code for objects>
  <PDDL code for initial state>
  <PDDL code for goal specification>
)
```

- **<problem name>** is a string that identifies the planning task, e.g. reachability-problem.
- **<domain name>** must match the domain name in the corresponding domain file, e.g., petri-net .



## ■ Planning.Domains

- URL: <http://planning.domains/>

# Planning.Domains

*A collection of tools for working with planning domains.*

planning.domains

:

1) api.planning.domains [🔗](#)

2) solver.planning.domains [🔗](#)

3) editor.planning.domains [🔗](#)

4) education.planning.domains [🔗](#)



# Example: The reachability problem

- **Objects:** Places and Transitions. For the specific case study, 6 places and 6 transitions.
- **Predicates:** Is a place an input place (or an output place) of a transition? Does a place contain a token?
- **Functions:** `total-cost` to keep track the cost of the plan under construction.
- **Actions/Operators:** Firing of a transition  $t$ .
- **Initial state:** A token is in place `start`. The other places do not contain any token.
- **Goal specification:** A token is in place `end`. The other places do not contain any token.



# The Reachability problem in PDDL Planning Domain

Objects of the domain and predicates describe the state of the world.

```
(define (domain petri-net)
  (:types place transition)
  (:predicates (token ?p - place)
    (input_place ?t - transition ?p - place)
    (output_place ?t - transition ?p - place))
  (:action fire
    :parameters (?t - transition)
    :precondition (forall (?p - place)
      (imply (input_place ?t ?p)
        (token ?p)))
    :effect (and (forall (?p - place)
      (when (input_place ?t ?p)
        (not (token ?p))))
      (forall (?p - place)
        (when (output_place ?t ?p)
          (token ?p)))
      (increase (total-cost) 1))))
```

Actions are described in terms of **preconditions** under which an action can be executed, and **effects** on the state of the world, stated in terms of the predicates.

)



# The Reachability problem in PDDL Planning Problem

```
(define (problem pr1)
  (:domain petri-net)
  (:objects start p1 p2 p3 p4 end - place
            a b c d e f - transition)
  (:init (token start) (input_place a start)
         (output_place a p1) (output_place a p2)
         (input_place b p1) (output_place b p3)
         (input_place c p2) (output_place c p4)
         (input_place d p1) (input_place d p2)
         (output_place d p3) (output_place d p4)
         (input_place e p3) (input_place e p4)
         (output_place e end) (input_place f p3)
         (input_place f p4) (output_place f p2)
         (= (total-cost) 0))
  (:goal (and (token end) (not (token start))
              (not (token p1)) (not (token p2))
              (not (token p3)) (not (token p4))))
  (:metric minimize (total-cost)))
)
```

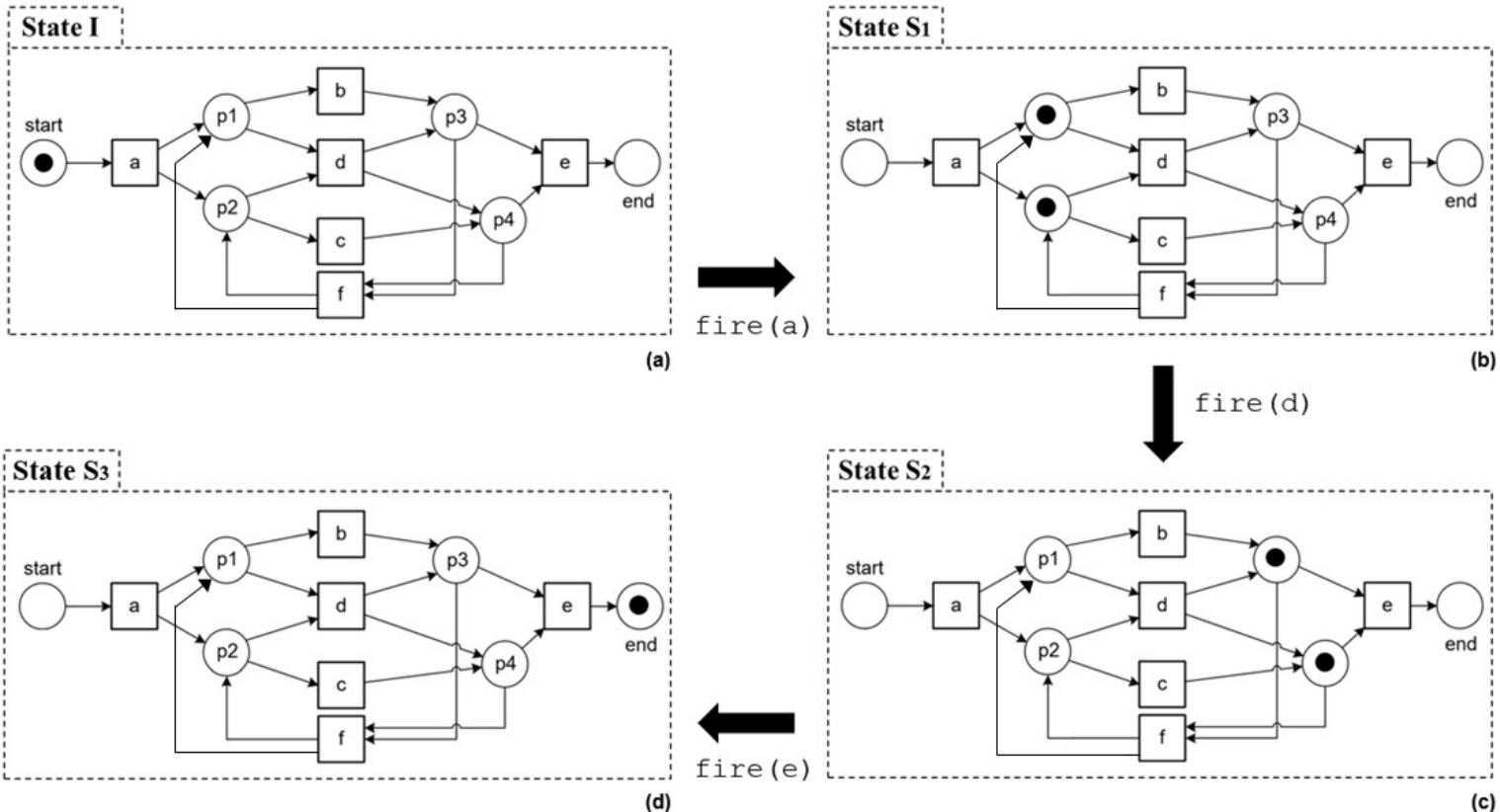


IBM

# The Reachability problem in PDDL

## Optimal plan

Begin plan  
1. (fire a)  
2. (fire d)  
3. (fire e)  
End plan



Since S<sub>3</sub> is a state satisfying G, the solution found is a **valid plan**.



# The Blocks World in PDDL

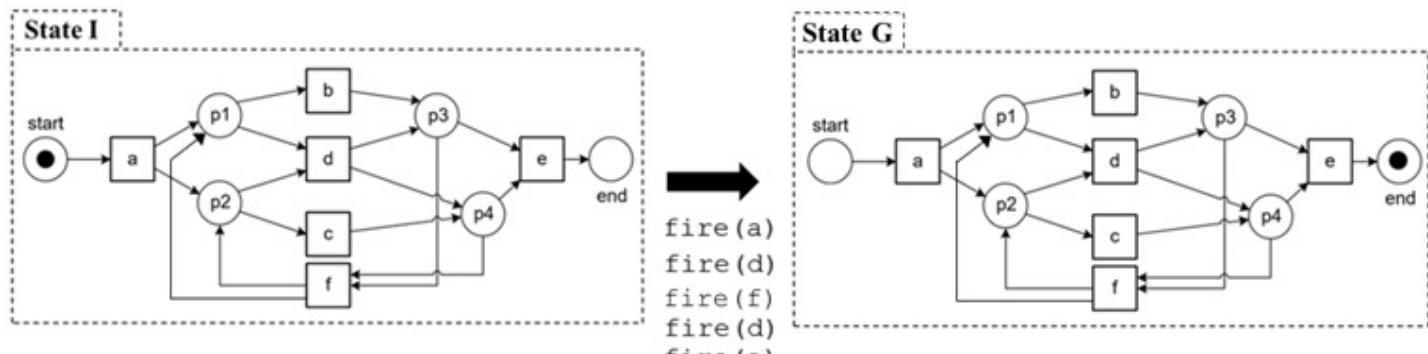
## *SubOptimal Plan*

The **quality of a solution** depends by the specific search algorithm employed by the planner.

Begin plan

1. (fire a)
2. (fire d)
3. (fire f)
4. (fire d)
5. (fire e)

End plan



For classical planning,  
the general problem of  
coming up with a plan is  
**NP-hard**



# Challenge of automated planning

- **Challenge:** achieving both **generality** and **scalability**.
  - **Generality:** A planner can solve *arbitrary problem instances*.
    - A planner does not know what the actions, and domain stand for.
    - This is very different from writing a *domain-specific solver*.
  - **Scalability:** Planners embed very effective **domain-independent heuristics** to drive the searching task towards the goal.
    - An *heuristic function* provides an estimate of the cost to reach the goal from the current state (Examples: Best-First Search, A\*, Hill Climbing, etc).
- State-of-the art planners\*\* provide **customized implementations** of the search algorithms with different properties of completeness, optimality, and memory complexity.

\*\*Cf. <http://icaps-conference.org/index.php/main/competitions>



- Several extensions of PDLL:

- **PDDL 1.2:** Base version of the language. Among the basic constructs, it includes **STRIPS**, **ADL** and **conditional effects**.
- **PDDL 2.1:** It introduces **numeric fluents** (e.g., to model non-binary resources such as time, distance, weight, etc.), **plan-metrics** (to allow *quantitative evaluation* of plans, and not just goal-driven), and **durative/continuous actions** (which could have variable, non-discrete length, conditions and effects).
- **PDDL 2.2:** It introduces **derived predicates** (to model the dependency of given facts from other facts), and **timed initial literals** (to model exogenous events occurring independently from plan-execution).
- **PDDL 3.0:** It introduces **preferences** (hard- and soft-constraints, in form of logical expressions, to be satisfied in specific points of the plan).
- **PDDL 3.1:** It introduces **object fluents** (functions' range can be any object-type).



1. New trends in Business Process Management
2. Basics of Automated Planning
3. **Automated Planning along the BPM Life-cycle**
  - Declarative modeling
  - Trace alignment against declarative process models
  - Interpretability
4. Conclusions

# BPM 2021 Tutorial Part II

# Declarative Modeling

Tathagata Chakraborti  
IBM Research

In collaboration with [Yara Rizk](#) · [Sarah Sreedharan](#) · [Yasaman Khazaeni](#) · [Christian Muise](#) · [Shubham Agarwal](#) · [Scott Boag](#) · [Vatche Ishakian](#) · [Dario Silva Moran](#) · [Rania Khalaf](#) · [Vinod Muthusamy](#) · [Merve Unuvar](#) · [Luis Lastras](#) · [Ondrej Bajgar](#) · [Arunima Chaudhary](#) · [Abhisekh Bhandwalder](#) · [Falk Pollock](#) · [Josef Ondrej](#) · [Miroslav Vodolan](#) · [Charlie Wiecha](#) · [Subbarao Kambhampati](#)

[IBM Research](#) · [Arizona State University](#) · now at [Queen's University](#) · now at [Future of Humanity Institute](#)

# Declarative Modeling = Scale Up

1. Scale up means things of same complexity require much smaller specification.
2. Scale up means specification of same size yield things of much higher complexity.

# What does a plan look like?

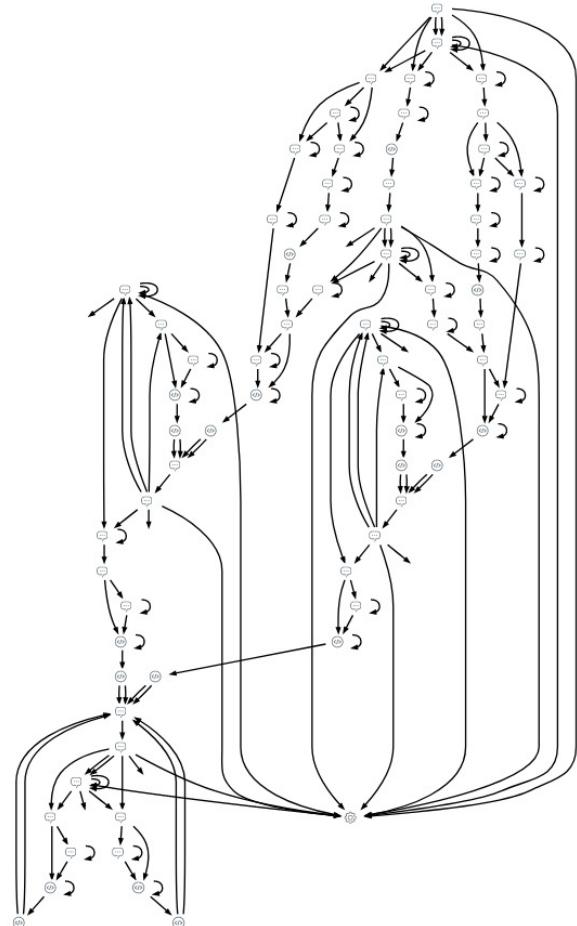
An online plan is a sequence.

Plan for most optimistic outcome and re-plan if that doesn't materialize at runtime.

```
Step 0: auth_agent ( loan_agent )
Step 1: slot_fill_agent ( loan_amount )
Step 2: slot_fill_agent ( account_number )
Step 3: slot_fill_agent ( credit_score )
Step 4: slot_fill_agent ( income )
Step 5: slot_fill_agent ( valid_id )
Step 6: loan_agent ( account_number,
income, credit_score, valid_id,
loan_amount )
```

An offline “contingent” plan is a graph or tree.

Plan for all outcomes and determine which one happened at runtime.



# What does a plan look like?

An online plan is a sequence.

Plan for most optimistic outcome and re-plan if that doesn't materialize at runtime.

```
Step 0: auth_agent ( loan_agent )
Step 1: slot_fill_agent ( loan_amount )
Step 2: slot_fill_agent ( account_number )
Step 3: slot_fill_agent ( credit_score )
Step 4: slot_fill_agent ( income )
Step 5: slot_fill_agent ( valid_id )
Step 6: loan_agent ( account_number,
income, credit_score, valid_id,
loan_amount )
```

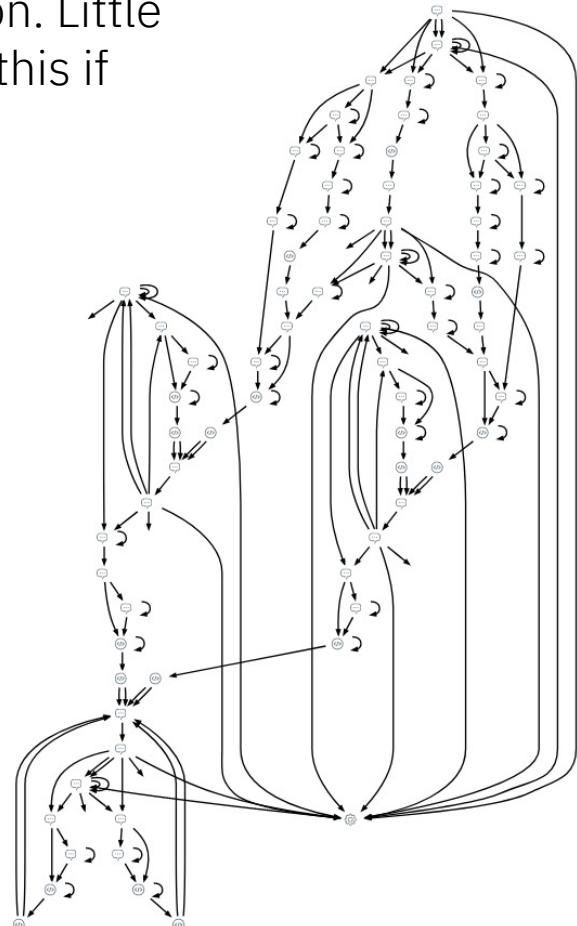
**Runtime:** Plan and re-plan. Easy to adapt to non-stationary domains.

An offline “contingent” plan is a graph or tree.

Plan for all outcomes and determine which one happened at runtime.

**Runtime:** Outcome determination. Little to no reasoning. No point doing this if domain is not stationary.

**Bonus:** You can see what it is you have made.



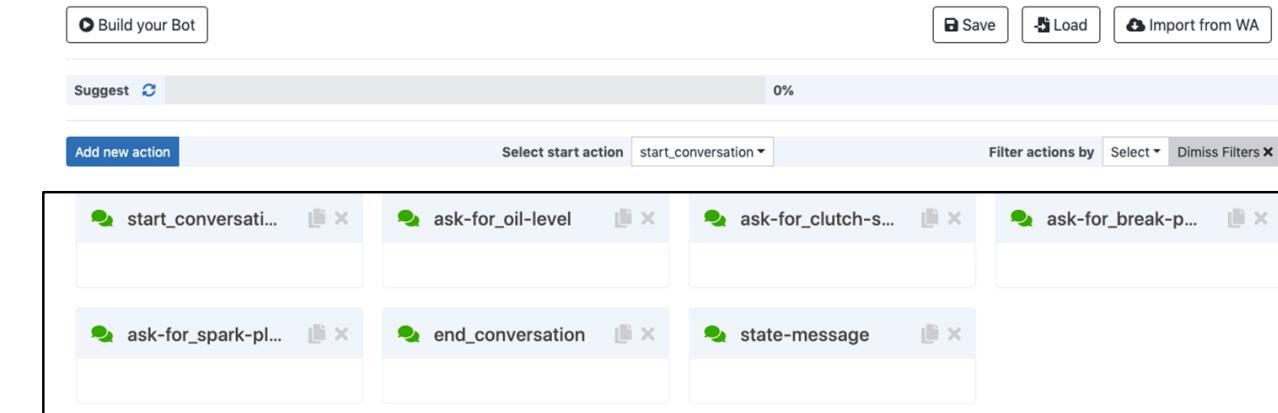
# Declarative Modeling

## Case Study 1 Goal Oriented Conversational Agents

Not chit chat, goal oriented e.g., customer support.  
They have underlying business processes governing  
the flow and/or outcome of conversation.

Involves manually building and maintaining a giant  
dialogue tree. E.g., Watson Assistant, Dialogflow.

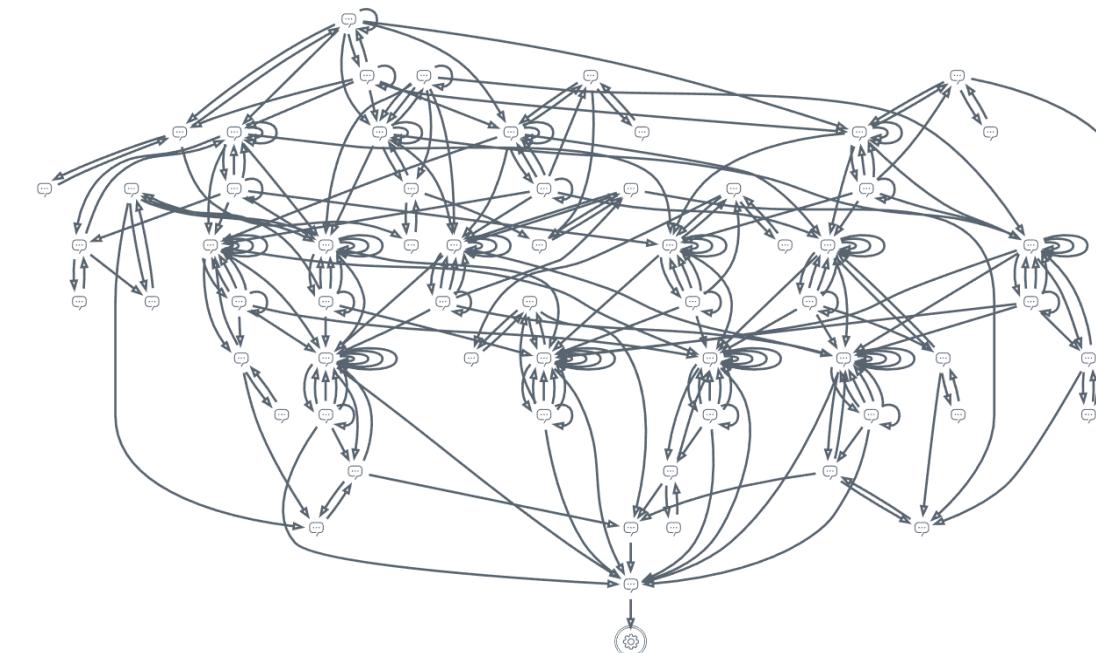
# Scale-up using declarative specification



7 atomic actions

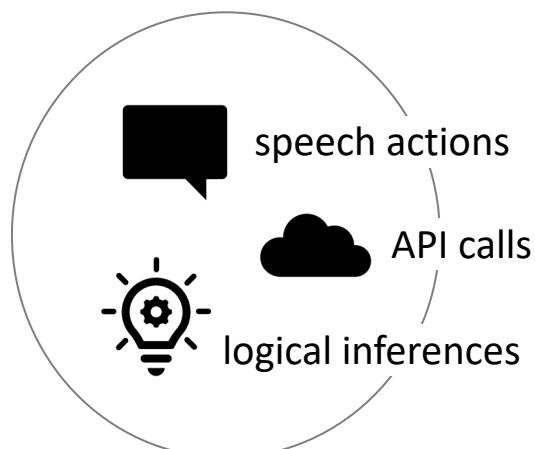


Composed process

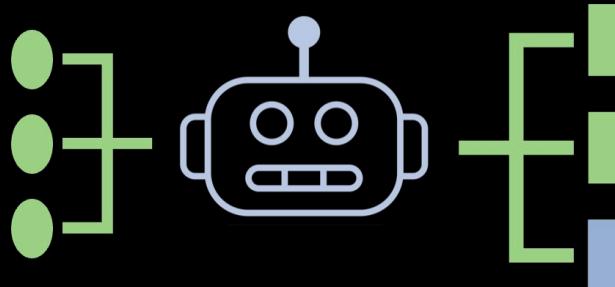


# Under the hood

## Anatomy of a declarative specification



The agent needs to know conditions that an action depends on.

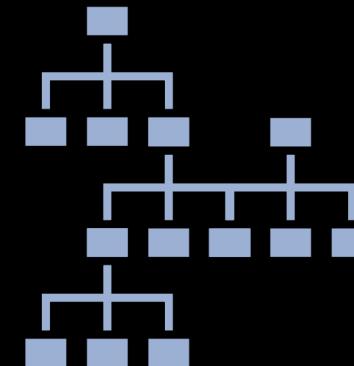


An action can have several possible outcomes.

Any one of the possible outcomes can occur during execution.

Each outcome impacts the task differently.

The domain authoring interface provides an abstraction to the industry standard PDDL language into which the specification is compiled to in the backend



A non-deterministic planner in the background composes decision tree automatically.

I: Oil level is good

A: Oil level is good, noted.

I: What is next?

A: Please check *clutch seal tightness*.

I: What are the options?

A: Tight, loose, needs repair.

I: It's loose.

A: Ok, clutch seal is loose.

A: What is the *status of spark plugs*?

I: Where do I find them?

A: Check the status of spark plugs at the upper part of the engine.

I: Ok.

I: Spark plugs pass.

A: Great spark plugs pass.

A: Please check *break pads*.

I: I'll take it from here.

A: As you wish.

I: What?

A: I'm waiting for you to report inspection items.

I: Ok.

I: Break pads are ok.

A: Ok, break pads pass.

Maintain context over prolonged interactions

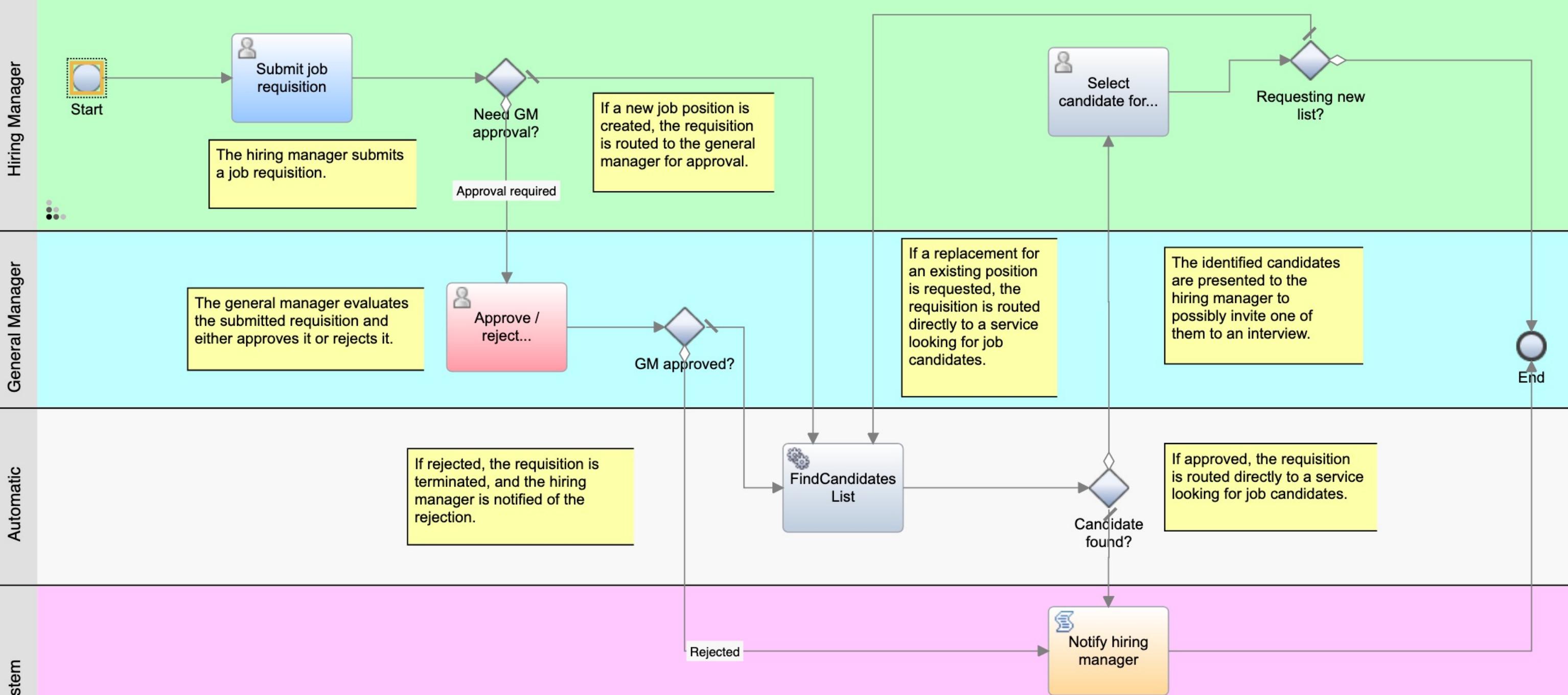
Log / connect to system actions

Initiative exchange between the agent and the user

Execution monitoring

Demo

# Hiring Process from IBM Workflow Center



# Specification in D3WA (Sample Hiring Process)

## Process Elements

Individual tasks defined in terms of inputs and outputs, either by the domain author or imported from existing processes

Bot Name Bob

Build your Bot

Save Load Import from WA Import from BPM

List of Variables Add new variable

gm\_approved flag

gm\_approval\_done flag

escalate flag

position json

candidates json

interview\_done flag

candidate\_success flag

List of Intents

WA Credentials

Suggest 0%

Add new action Select start action submit\_position\_request Filter actions by Select Dimiss Filters

notify\_hiring\_m...

find\_position\_ca...

candidate\_found

select\_candidat...

request\_new\_can...

gm\_approval

is\_new\_position

escalate

submit\_position...

# Composed Process in D3WA

## Goal Conditions

Provide goal conditions of the desired process (when the hiring manager is notified or candidate is identified).

**Cloudfunction** notify\_hiring\_manager

Call URL <https://us-south.functions.cloud.ibm.com/api/v1/web/C>

```
{ "outcomes": [ { "outcome_chosen": "notify_hiring_manager" } ] }
```

Call Parameters

**Generate Code** **Simulate Endpoint**

**Add statuses of variables needed for this action.**

NEED interview\_done

**Enter possible outcomes of this action.**

**Enter updates that will happen in all outcomes.**

**Variables**  
Comma separated variable list defined by endpoint:  
 End conversation here?

**Updates to variables as a result of this outcome**

**Logic** request\_new\_candidate

**Add statuses of variables needed for this action.**

NEED interview\_done

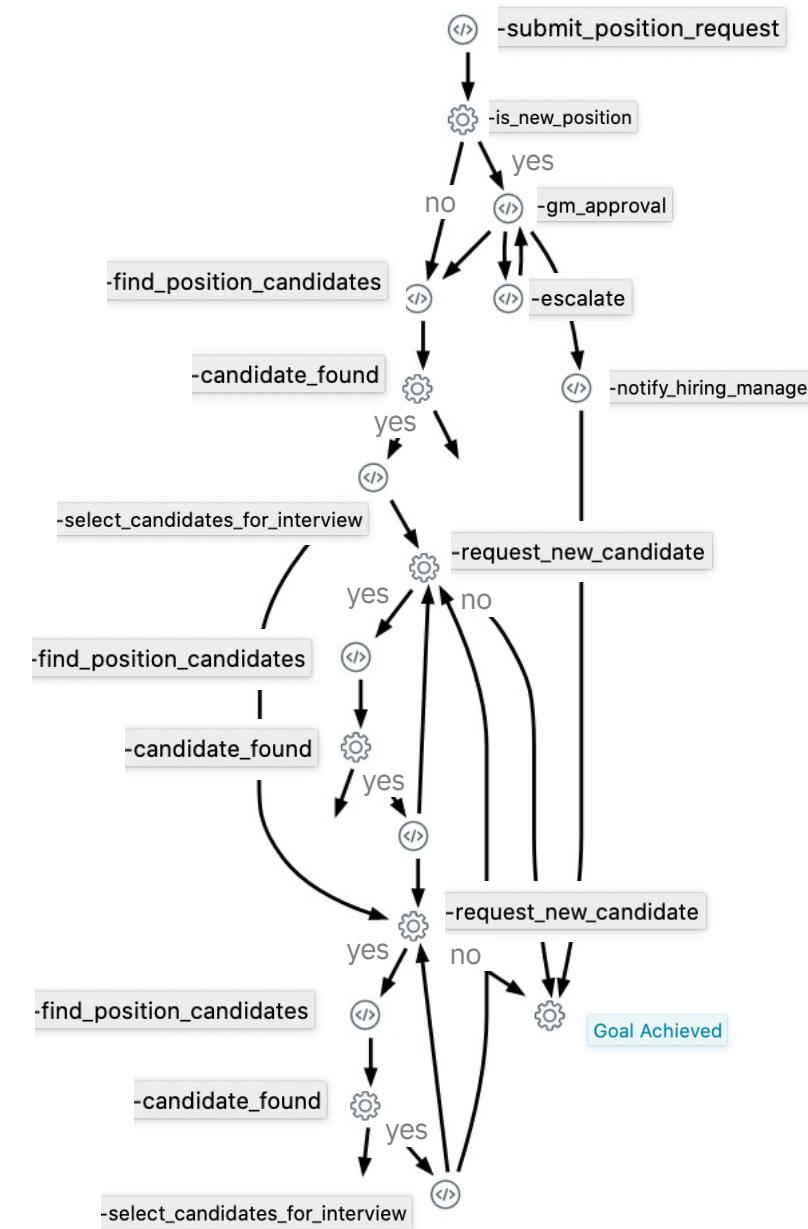
**Enter possible outcomes of this action.**

**Enter updates that will happen in all outcomes.**

**Trigger**  
Boolean condition to trigger outcome.  
 End conversation here?  
 Follow-up this outcome?  
Follow up with **find\_position\_candidates**

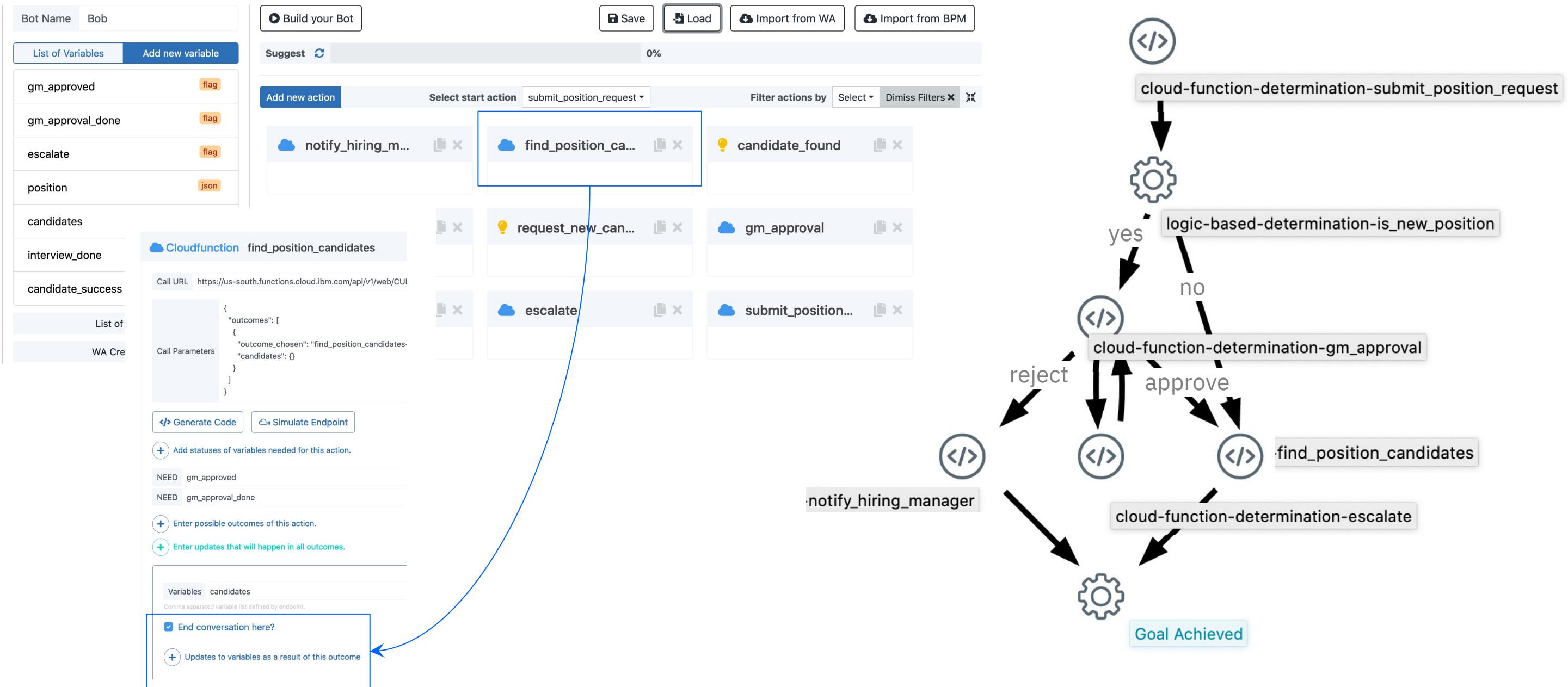
**Updates to variables as a result of this outcome**

**Trigger** candidate\_success == true  
Boolean condition to trigger outcome.  
 End conversation here?



# Process Modification

Same as before + ask the process to end right after finding candidates



# Declarative Modeling

## Case Study 2 Goal Oriented Conversation meets Web Service Composition

Optimizing business processes for automation.

Involves manually building and maintaining a giant dialogue tree. E.g., Watson Assistant, Dialogflow.

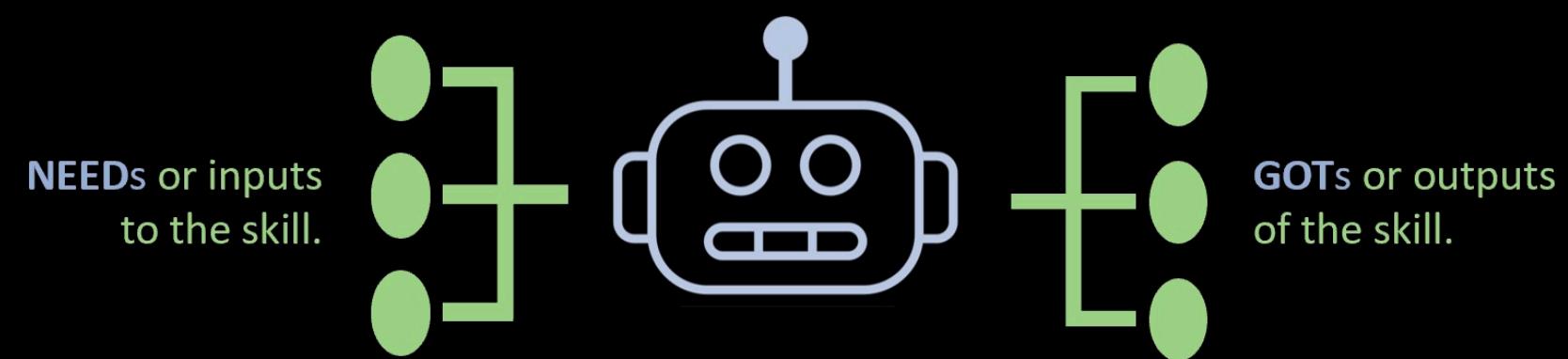
# Web Service Composition using Automated Planning

New web services =  $\sum$  existing web services

An age-old application of automated planning: chain together services by composing their INs and OUTs to get macro services.

These services can be used as the [units of automation](#) in authoring business processes as well.

# Web Service Composition using Automated Planning

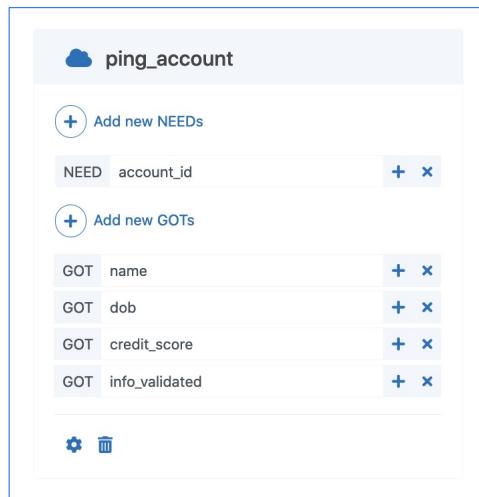


# D3BA Process Composition Example

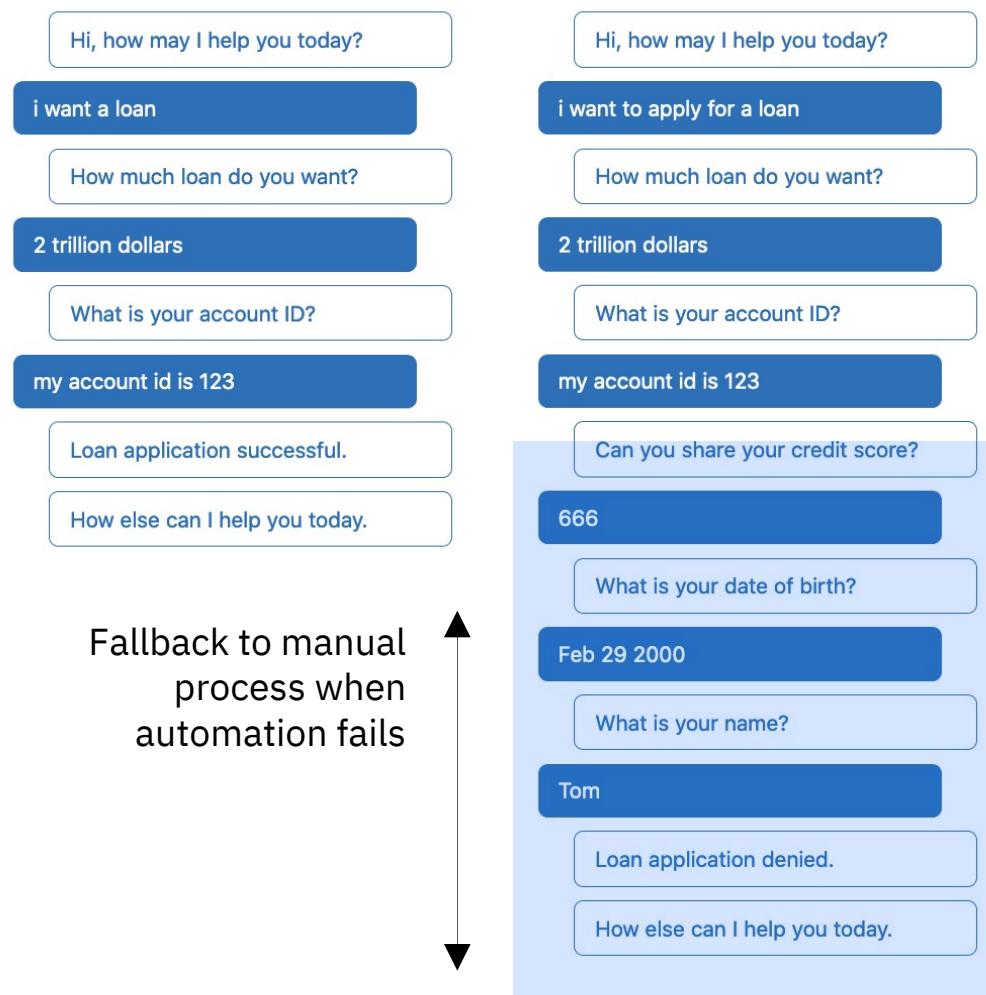
Business Process + Catalog of Skills = Optimized Business Process

Optimization in terms of maximization of automation opportunities

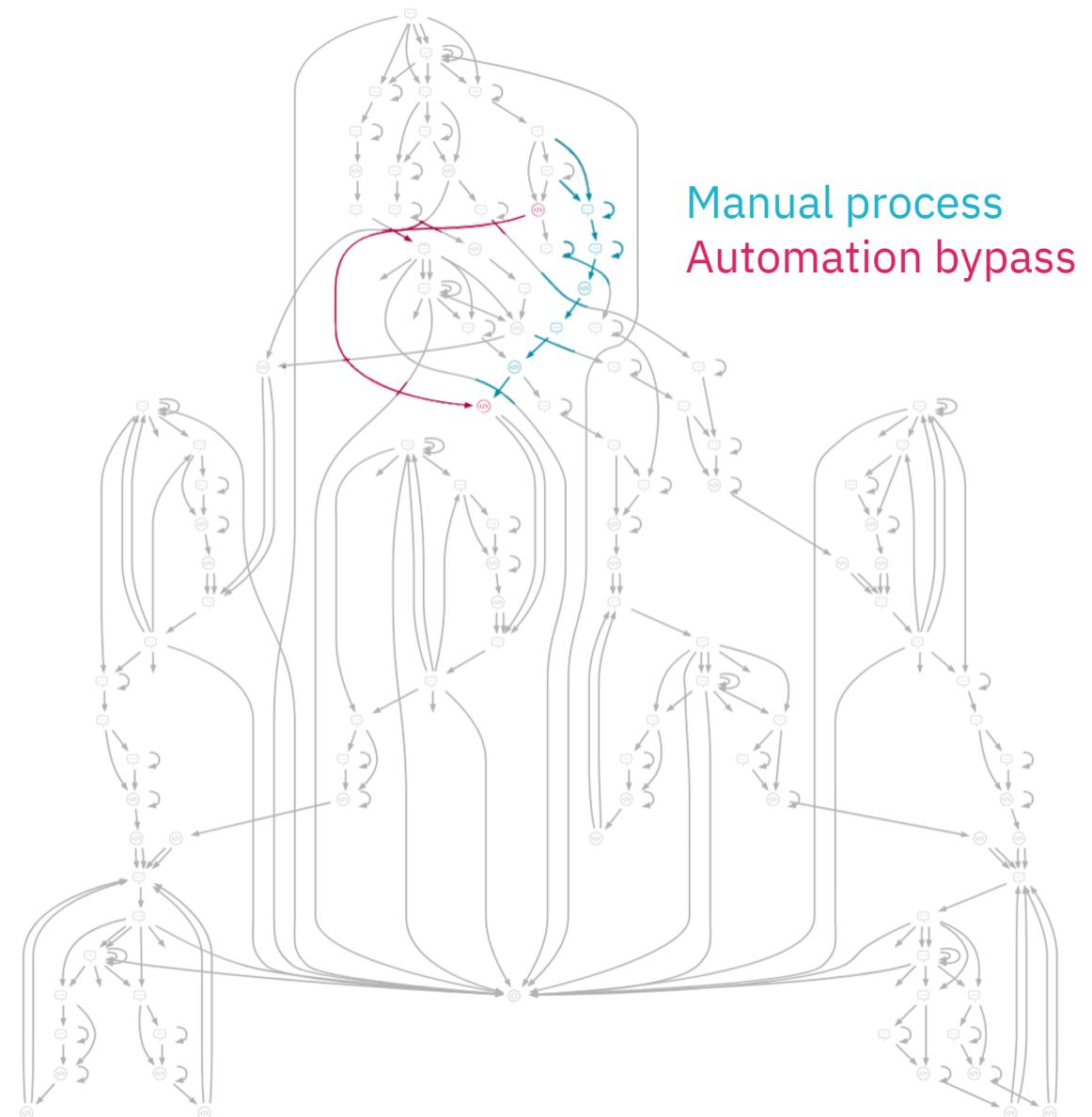
Fallbacks to manual process when automation fails



A **skill** is the unit of automation



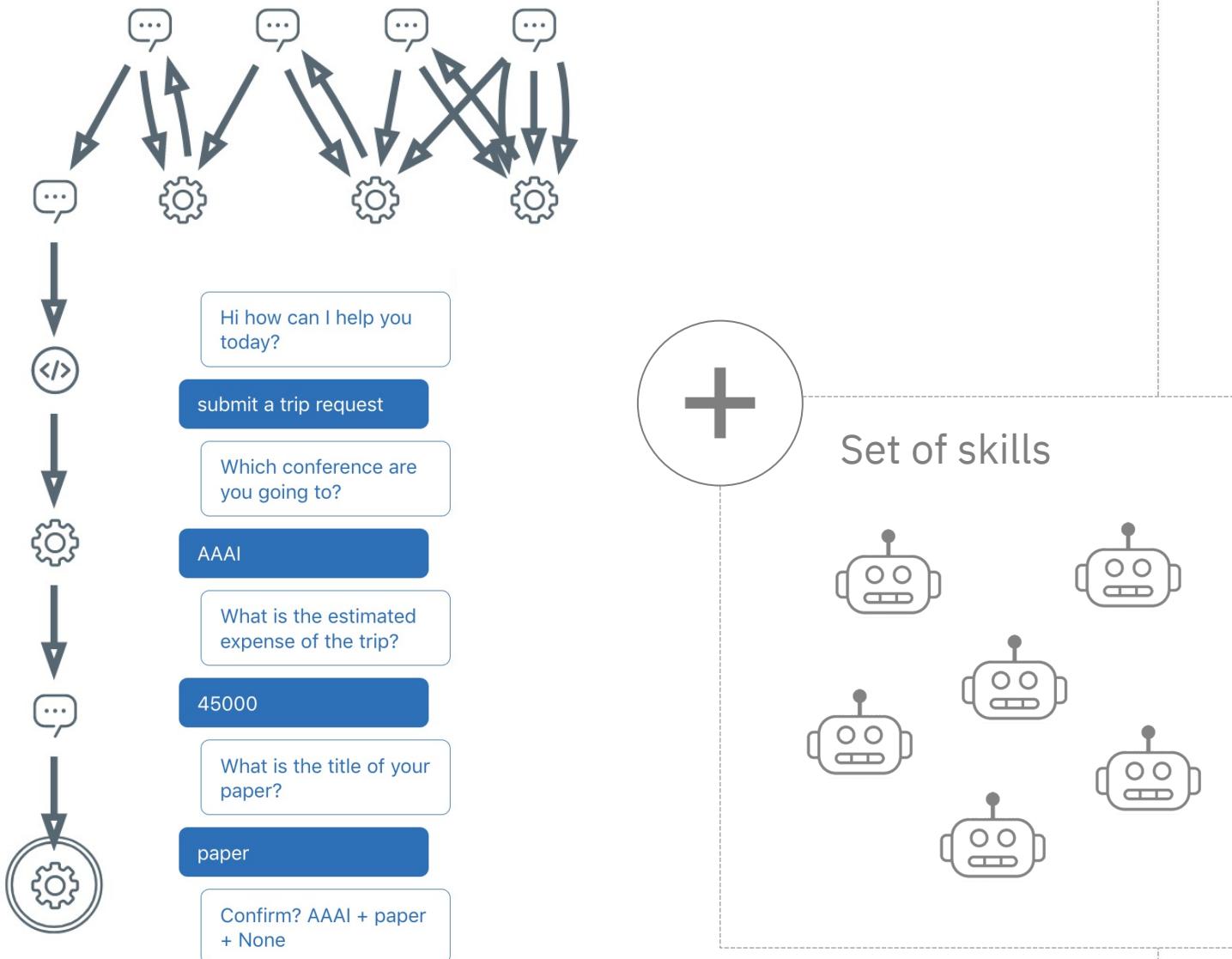
Fallback to manual process when automation fails



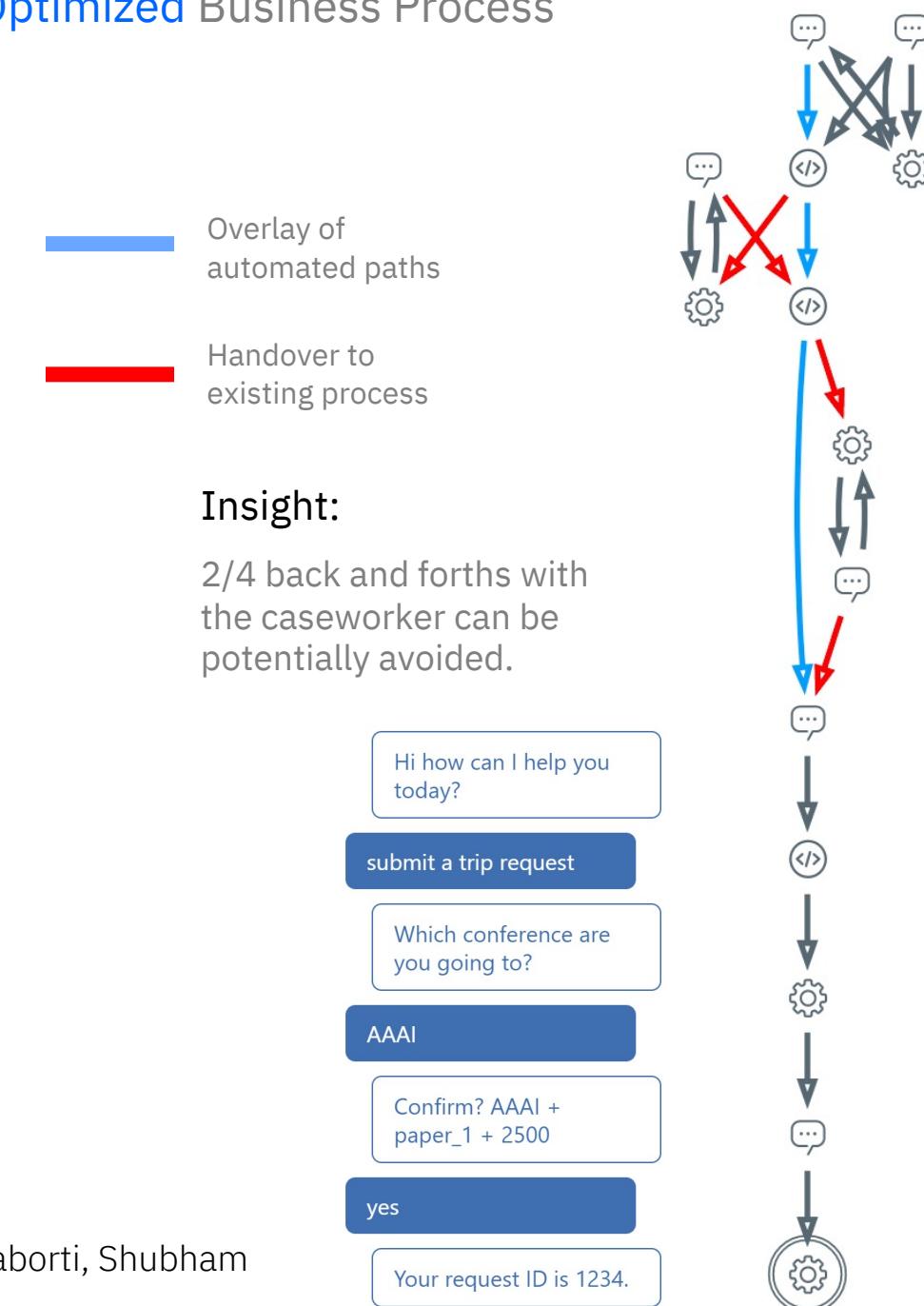
Manual process  
Automation bypass

# D3BA Process Composition Example

Example of an existing Business Process



Optimized Business Process



Overlay of automated paths

Handover to existing process

## Insight:

2/4 back and forths with the caseworker can be potentially avoided.

Demo

# References

[A Unified Conversational Assistant Framework for Business Process Automation.](#) Yara Rizk, Abhisekh Bhandwalder, Scott Boag, Tathagata Chakraborti, Vatche Isahagian, Yasaman Khazaeni, Falk Pollock, and Merve Unuvar. AAAI 2020 Workshop on Intelligent Process Automation.

[D3WA+ -- A Case Study of XAIP in a Model Acquisition Task for Dialogue Planning.](#) Sarath Sreedharan, Tathagata Chakraborti, Christian Muise, Yasaman Khazaeni, and Subbarao Kambhampati. ICAPS 2020.

[Planning for Goal-Oriented Dialogue Systems.](#) Christian Muise, Tathagata Chakraborti, Shubham Agarwal, Ondrej Bajgar, Arunima Chaudhary, Luis A. Lastras-Montano, Josef Ondrej, Miroslav Vodolan, and Charlie Wiecha. Technical Report (2019).

[D3BA: A Tool for Optimizing Business Processes Using Non-Deterministic Planning.](#) Tathagata Chakraborti, Shubham Agarwal, Yasaman Khazaeni, Yara Rizk, and Vatche Isahagian. BPM 2020 Workshop on AI4BPM.

[From Robotic Process Automation to Intelligent Process Automation: Emerging Trends.](#) Tathagata Chakraborti, Vatche Ishakian, Rania Khalaf, Yasaman Khazaeni, Vinod Vinod Muthusamy, Yara Rizk, and Merve Unuvar. BPM 2020 RPA Forum.

[Explainable Composition of Aggregate Assistants.](#) Sarath Sreedharan, Tathagata Chakraborti, Yara Rizk, and Yasaman Khazaeni. ICAPS 2020 Workshop on Explainable AI Planning.

In collaboration with [Yara Rizk](#) · [Sarah Sreedharan](#) · [Yasaman Khazaeni](#) · [Christian Muise](#) · [Shubham Agarwal](#) · [Scott Boag](#) · [Vatche Ishakian](#) · [Dario Silva Moran](#) · [Rania Khalaf](#) · [Vinod Muthusamy](#) · [Merve Unuvar](#) · [Luis Lastras](#) · [Ondrej Bajgar](#) · [Arunima Chaudhary](#) · [Abhisekh Bhandwalder](#) · [Falk Pollock](#) · [Josef Ondrej](#) · [Miroslav Vodolan](#) · [Charlie Wiecha](#) · [Subbarao Kambhampati](#)



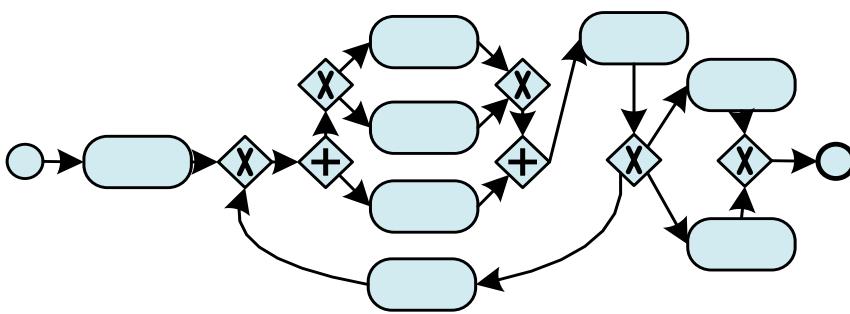
1. New trends in Business Process Management
2. Basics of Automated Planning
3. **Automated Planning along the BPM Life-cycle**
  - Declarative modeling
  - **Trace alignment against declarative process models**
  - Interpretability
4. Conclusions



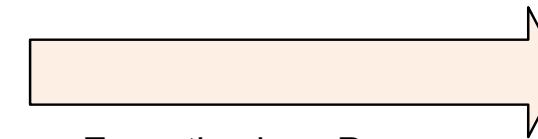
IBM

# Conformance Checking

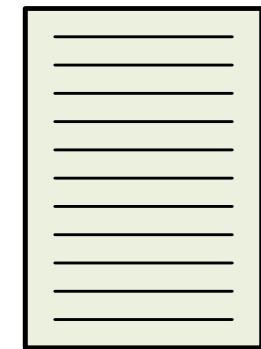
Any execution of a process model produces a new **execution trace** (i.e., a **process instance**) recorded in an **event log**.



process model



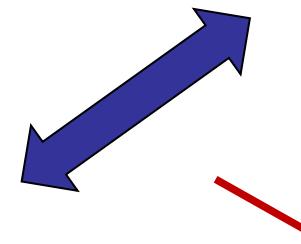
Execution by a Process Management System



event log



regulations



Concrete process executions  
are compliant with regulations  
and laws?



# The trace alignment problem

- Process models are typically not enforced by information systems (human behavior is often involved).
  - Traces can be **dirty**, with **spurious** or **missing events**.
- **Trace alignment** is the problem of **cleaning** such dirty traces against process models to the aim of:
  - verify if a trace is **compliant** with its underlying process model;
  - identifying the **root** and the **severity** of each deviation;
  - **repairing the trace** to make it compliant with the process model.
- The state-of-the-art solutions to compute optimal alignments
  - provide **ad-hoc implementations** of the A\* algorithm.
  - **do not scale efficiently** when process models and event logs are of considerable size.

**SOLUTION:** The problem of computing optimal alignments can be formulated as a **planning problem** in PDDL



- Given a trace  $t$  and a DECLARE model  $D$  (which is used to define the regulations) find the optimal alignment of  $t$  with respect to  $D$ .
  - A DECLARE model  $D = (A, \pi_D)$  consists of a set of activities  $A$  involved in a process and a collection of **temporal constraints**  $\pi_D$  defined over  $A$ .
  - DECLARE constraints (aka **templates**) define parameterized classes of properties and enjoy a precise semantics in  $LTL_f$  (LTL over finite traces).

### Existence(A)

LTL Formalization:  $\diamond A$

$A$  occurs at least 1 time.

*BCAAC ✓ BCC ✗*

### Absence(C)

LTL Formalization:  $\neg \diamond C$

$A$  never occur.

*BAA ✓ BCAC ✗*

### Response(A, B)

LTL Formalization:  $\square (A \rightarrow \diamond B)$

If  $A$  occurs, then  $B$  occurs after  $A$ .

*BCAAC ✗ CAACB ✓ BCC ✓*



IBM

# From LTL<sub>f</sub> to DFAs

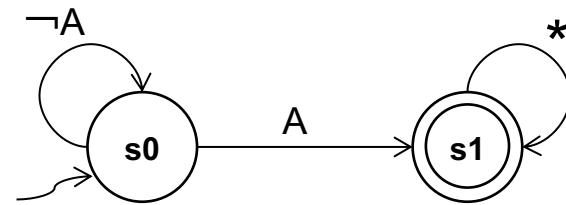
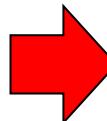
- For any LTL<sub>f</sub> formula there exists a DFA that accepts all the traces satisfying the formula.

## Existence(A)

LTL Formalization:  $\Diamond A$

A occurs at least 1 time.

BCAAC ✓ BCC ✗

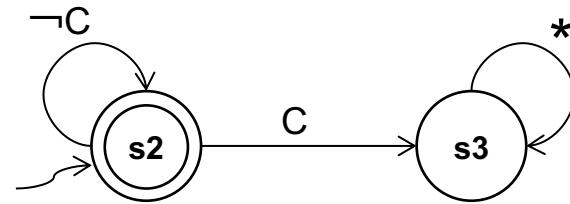
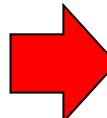


## Absence(C)

LTL Formalization:  $\neg \Diamond C$

A never occur.

BAA ✓ BCAC ✗

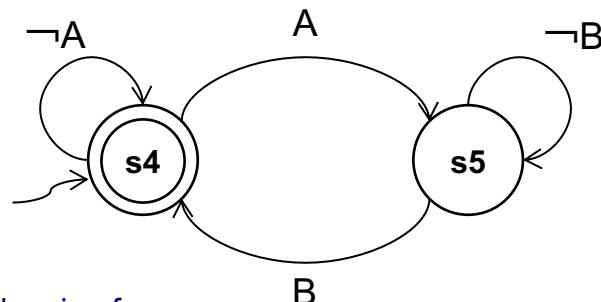
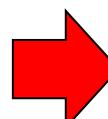


## Response(A, B)

LTL Formalization:  $\Box (A \rightarrow \Diamond B)$

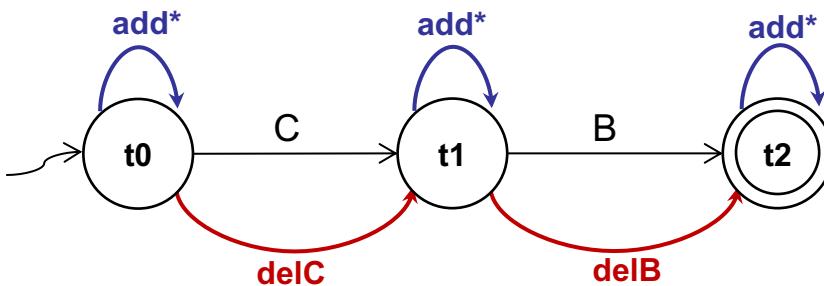
If A occurs, then B occurs after A.

BCAAC ✗ CAACB ✓ BCC ✓





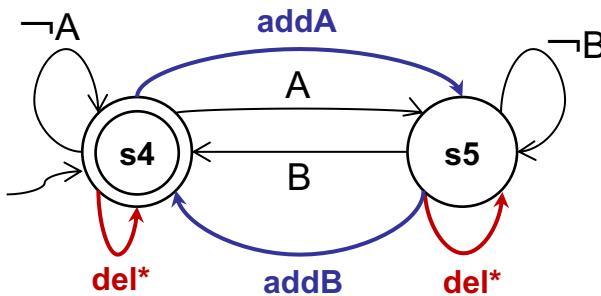
- Trace alignment can be solved using automata:
  - One automaton for the trace (**trace automaton**).



- Accepts input trace ( $\langle C, B \rangle$ ) plus all other traces, however...
- ...changes wrt. input trace must be marked by **add/del**, e.g.,
  - $\langle C, B, C \rangle = C B \text{ add}C$
  - $\langle B, C, B, B \rangle = \text{del}C B \text{ add}C \text{ add}B \text{ add}B$
- **Adds** and **dels** have (possibly different) positive costs.



- One automaton per constraint (**constraint automaton**) **augmented** to account for adds and dels.



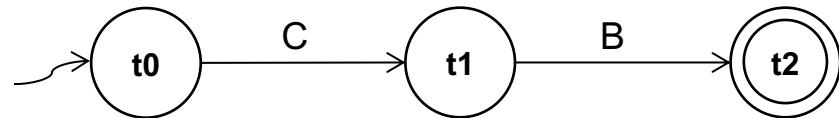
- Accepts all (possibly repaired) traces satisfying the constraint.
- An **alignment** is a sequence of **synchronous steps** performed in all *augmented constraint automata* and in the *augmented trace automaton* such that -- at the end of the alignment -- each automaton is **in at least one** accepting state.



IBM

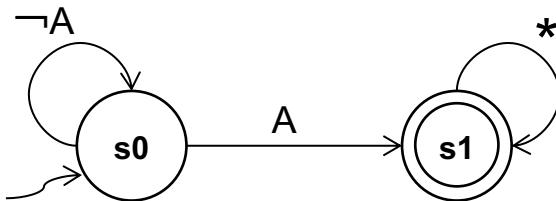
# An example of Trace Alignment

**Trace: <C,B>**

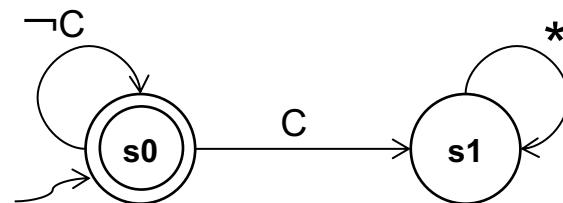


**LTL<sub>f</sub> Constraints**

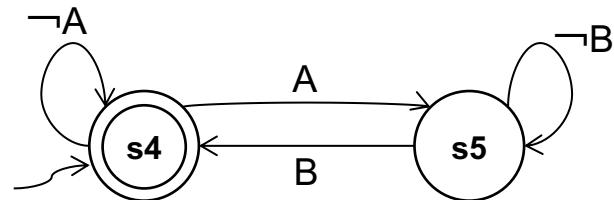
Existence(A):  $\Diamond A$



Absence(C):  $\neg \Diamond C$



Response(A,B):  $\Box(A \rightarrow \Diamond B)$





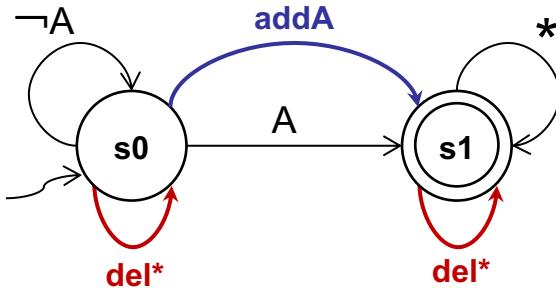
IBM

# An example of Trace Alignment

Trace:  $\langle C, B \rangle$

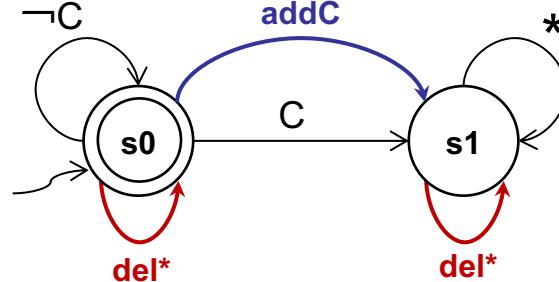
LTL<sub>f</sub> Constraints

Existence(A):  $\diamond A$

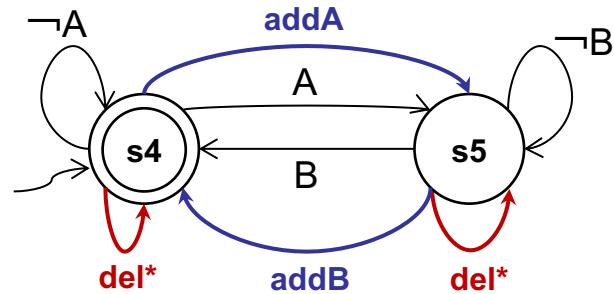


Augmented trace automaton  
and augmented constraint automata

Absence(C):  $\neg \diamond C$



Response(A,B):  $\square(A \rightarrow \diamond B)$





IBM

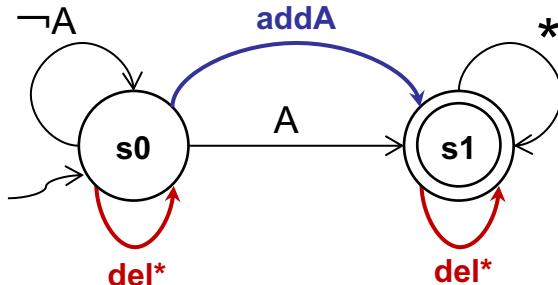
# An example of Trace Alignment

Trace:  $\langle C, B \rangle$

Optimal Plan:  $\langle \text{delC}, \text{addA}, B \rangle$

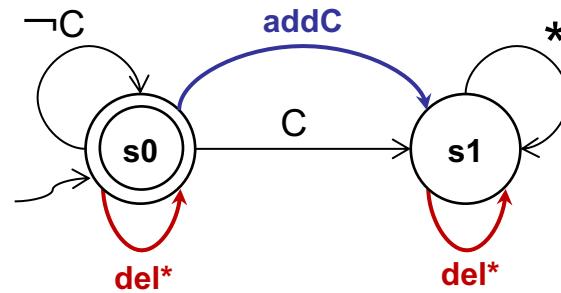
LTL<sub>f</sub> Constraints

Existence(A):  $\Diamond A$

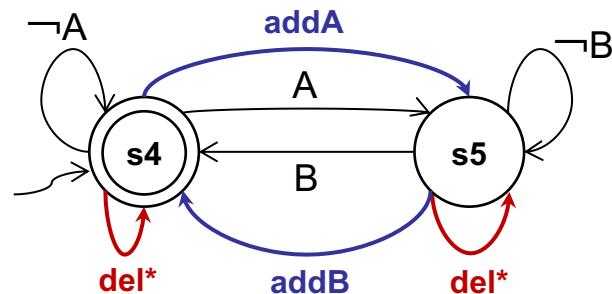


...if adds and dels have unitary cost.

Absence(C):  $\neg \Diamond C$



Response(A,B):  $\Box(A \rightarrow \Diamond B)$





- The automata-based approach can be recast as a **cost-optimal planning problem** using PDDL.
  - **Planning Domain:**
    - Input events modeled by **synchronization actions** with null cost.
    - **Adds** and **dels** modeled by planning actions with positive costs.
    - **Domain propositions** encode the structure and the dynamics of the augmented trace and of all augmented constraint automata.
  - **Problem:**
    - **Initial state:** all automata in their starting state.
    - **Goal state:** all automata in (at least one) final state.
  - **Solution:**
    - **Optimal** (i.e., **minimal-cost**) **plan** to reach the goal state.



# PDDL Planning Domain

## *Boolean Predicates*

```
(:types trace_state automaton_state - state activity)
```

They identify the states of any constraint automaton and of the trace automaton.

It captures the activities involved in a transition between two states of a constraint/trace automaton.

```
(:predicates
```

```
(trace ?t1 - trace_state  
      ?e - activity  
      ?t2 - trace_state)
```

```
(automaton ?s1 - automaton_state  
          ?e - activity  
          ?s2 - automaton_state)
```

```
(cur state ?s - state)
```

```
(final state ?s - state)
```

They hold if there exists a transition in the trace/constraint automaton from two states, being  $e$  the activity involved in the transition.

They hold if  $s$  is the current/accepting state of a trace/constraint automaton.

```
)
```



# PDDL Planning Domain

## Sync action

It is applied only if there exists a transition from the current state  $t_1$  of the trace automaton to a subsequent state  $t_2$ , being  $e$  the activity involved in the transition. The action **has no cost**, as it stands for no change in the trace.

```
(:action sync
:parameters (?t1 - trace_state ?e - activity
             ?t2 - trace_state)
:precondition (and (cur_state ?t1) (trace ?t1 ?e ?t2))
:effect (and (not (cur_state ?t1)) (cur_state ?t2)
             (forall (?s1 ?s2 - automaton_state)
                     (when (and (cur_state ?s1)
                                (automaton ?s1 ?e ?s2))
                         (and (not (cur_state ?s1))
                             (cur_state ?s2)))))))
```

**CONDITIONAL EFFECT:** The action is performed in each constraint automaton for which there exists a transition involving the activity  $e$  that connects  $s_1$  – the current state of the automaton – with a different state  $s_2$ .



```
(:action add
:parameters (?e - activity)
:effect (and (increase (total-cost) 1)
(forall (?s1 ?s2 - automaton_state)
(when (and (cur_state ?s1)
(automaton ?s1 ?e ?s2))
(and (not (cur_state ?s1))
(cur_state ?s2))))))
```

Add actions make total cost of the alignment increasing of a predefined value.

**CONDITIONAL EFFECT:** The action is performed only for transitions involving the activity  $e$  between two different states of any constraint automaton, with the current state of the trace automaton that remains the same after the execution of the action.



Del actions make total cost of the alignment increasing of a predefined value.

```
(:action del
:parameters (?t1 - trace_state ?e - activity ?t2 - trace_state)
:precondition (and (cur_state ?t1) (trace ?t1 ?e ?t2))
:effect (and (increase (total-cost) 1)
              (not (cur_state ?t1)) (cur_state ?t2)))
```

It yields a single move in the trace automaton.



IBM

# Initial and Goal State in PDDL

```
(:objects  
t0 t1 t2 - trace_state  
s4 s5 - automaton_state  
A B C - activity)
```

```
(:init  
(= (total-cost) 0)
```

```
(cur_state t0)  
(trace t0 C t1)  
(trace t1 B t2)  
(final_state t2)
```

Representation of the trace automaton.

```
(cur_state s4)  
(automaton s4 A s5)  
(automaton s5 B s4)  
(final_state s5))
```

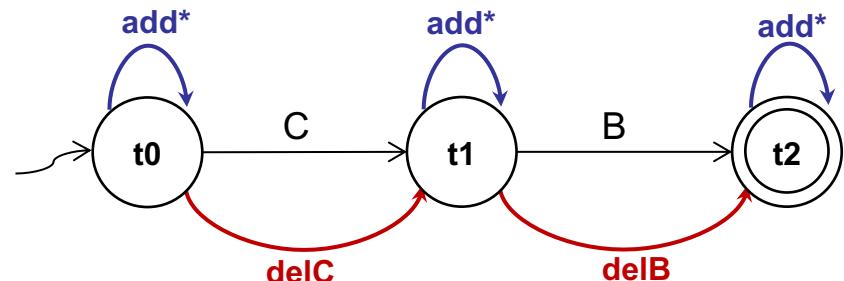
Representation of the constraint automaton.

```
(:goal (forall (?s - state)  
                 (imply (cur_state ?s) (final_state ?s))))
```

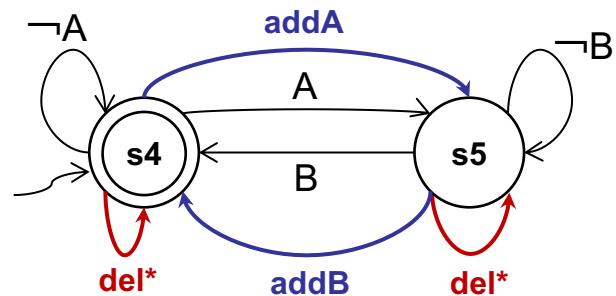
```
(:metric minimize (total-cost))
```

Minimization of the total cost of the alignment.

## Trace



□ ( $A \rightarrow \Diamond B$ ):





1. New trends in Business Process Management
2. Basics of Automated Planning
3. **Automated Planning along the BPM Life-cycle**
  - Declarative modeling
  - Trace alignment against declarative process models
  - **Interpretability**
4. Conclusions

BPM 2021 Tutorial Part III

# Interpretability

Tathagata Chakraborti

IBM Research

In collaboration with [Yara Rizk](#) · [Sarah Sreedharan](#) · [Yasaman Khazaeni](#) · [Christian Muise](#) · [Shubham Agarwal](#) · [Scott Boag](#) · [Vatche Ishakian](#) · [Dario Silva Moran](#) · [Rania Khalaf](#) · [Vinod Muthusamy](#) · [Merve Unuvar](#) · [Luis Lastras](#) · [Ondrej Bajgar](#) · [Arunima Chaudhary](#) · [Abhisekh Bhandwalder](#) · [Falk Pollock](#) · [Josef Ondrej](#) · [Miroslav Vodolan](#) · [Charlie Wiecha](#) · [Subbarao Kambhampati](#)

[IBM Research](#) · [Arizona State University](#) · now at [Queen's University](#) · now at [Future of Humanity Institute](#)

# Interpretability for the Domain Author

An online plan is a sequence.

Plan for most optimistic outcome and re-plan if that doesn't materialize at runtime.

```
Step 0: auth_agent ( loan_agent )
Step 1: slot_fill_agent ( loan_amount )
Step 2: slot_fill_agent ( account_number )
Step 3: slot_fill_agent ( credit_score )
Step 4: slot_fill_agent ( income )
Step 5: slot_fill_agent ( valid_id )
Step 6: loan_agent ( account_number,
income, credit_score, valid_id,
loan_amount )
```

Runtime: Plan and re-plan. Easy to adapt to non-stationary domains.

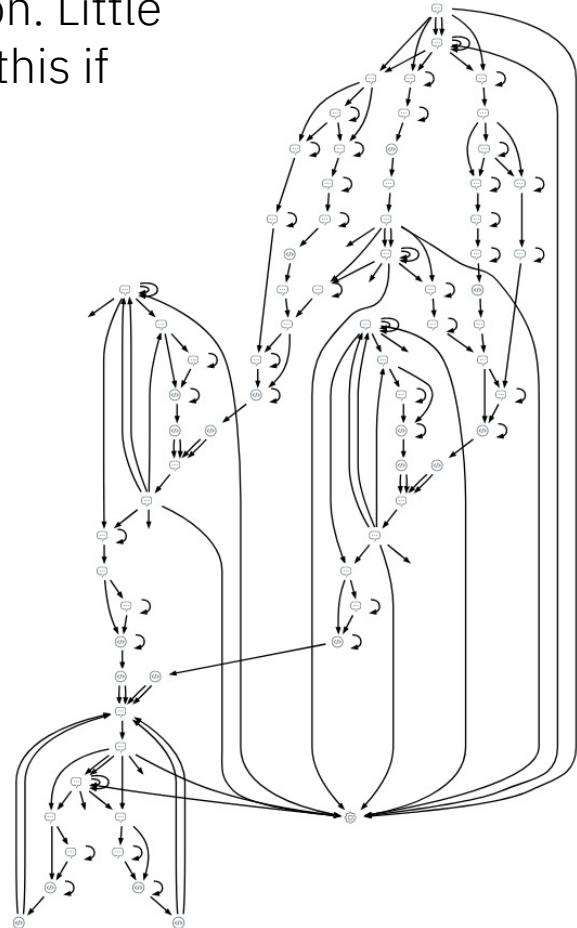
An offline “contingent” plan is a graph or tree.

Plan for all outcomes and determine which one happened at runtime.

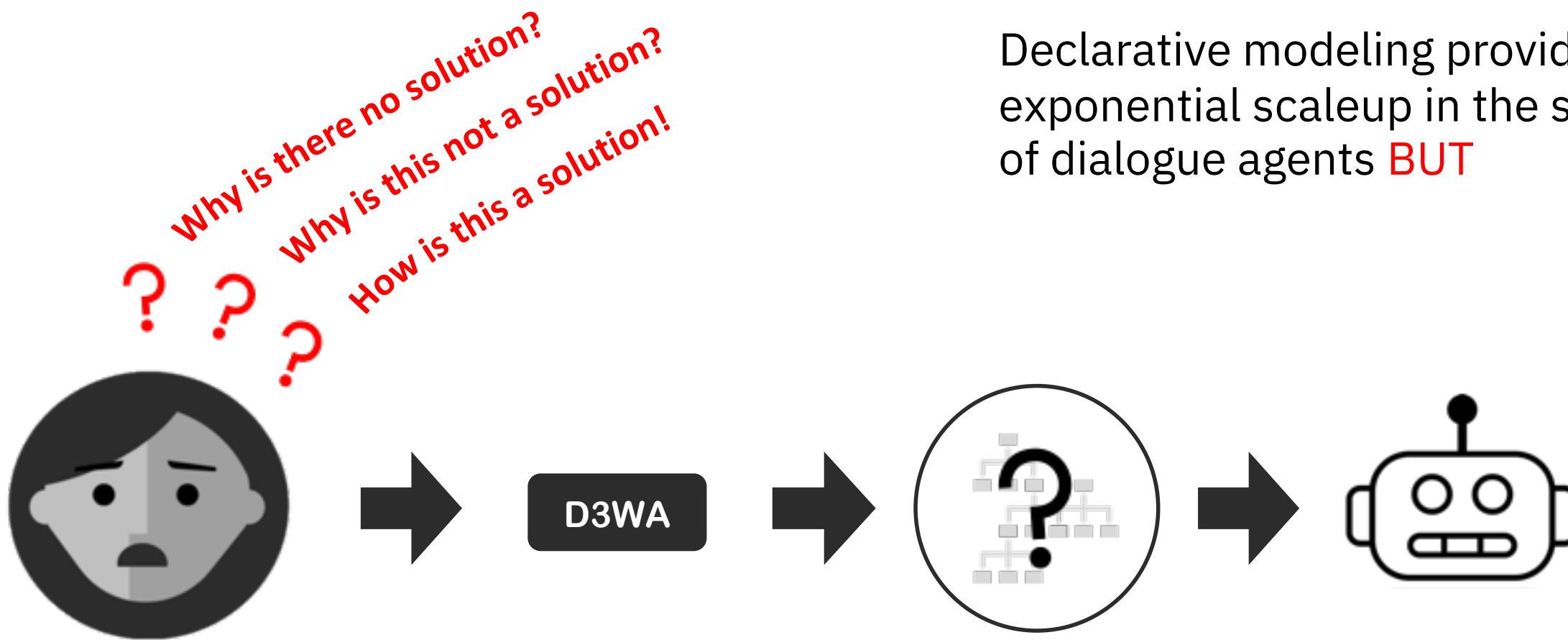
Runtime: Outcome determination. Little to no reasoning. No point doing this if domain is not stationary.

Bonus: You can see what it is you have made.

For the domain author.



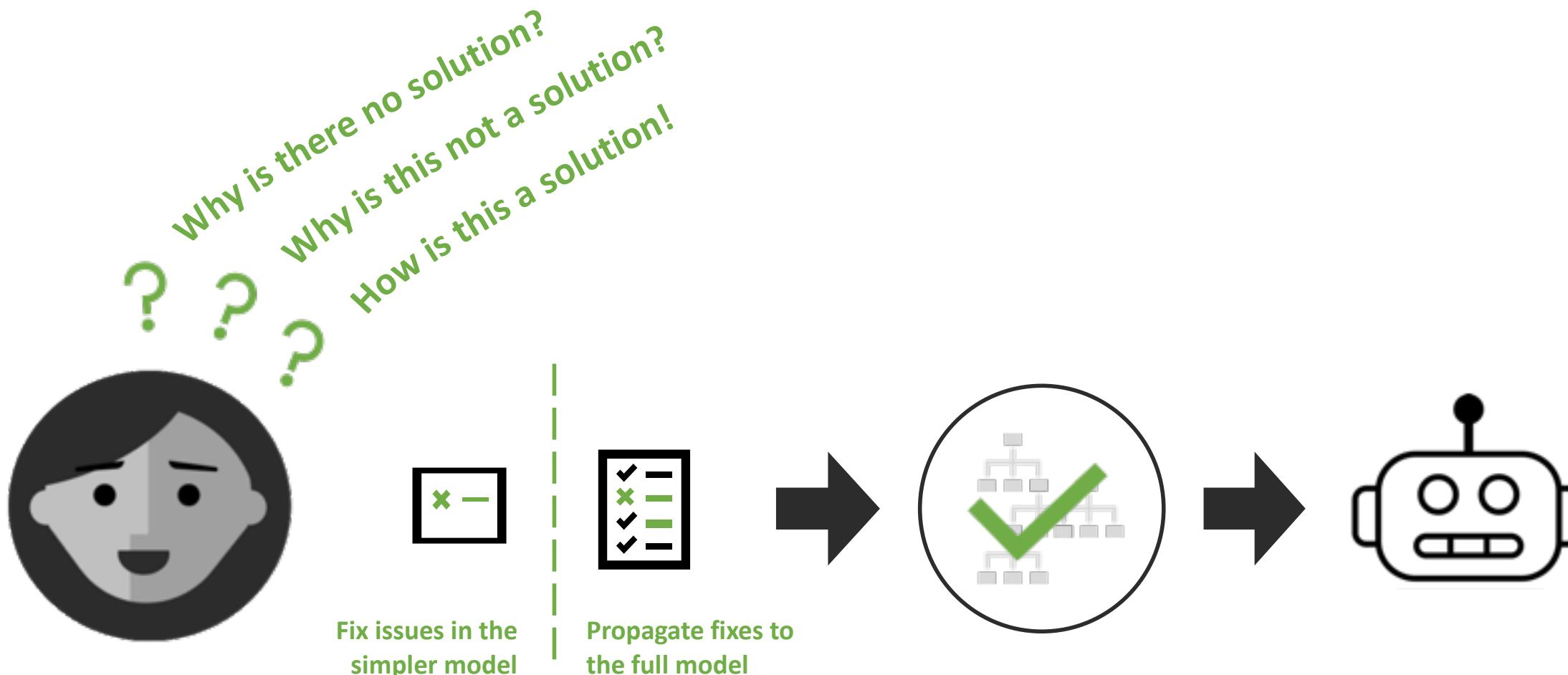
# Non-deterministic Planning as a design tool: Compose --> Debug



Declarative modeling provides exponential scaleup in the sophistication of dialogue agents **BUT**

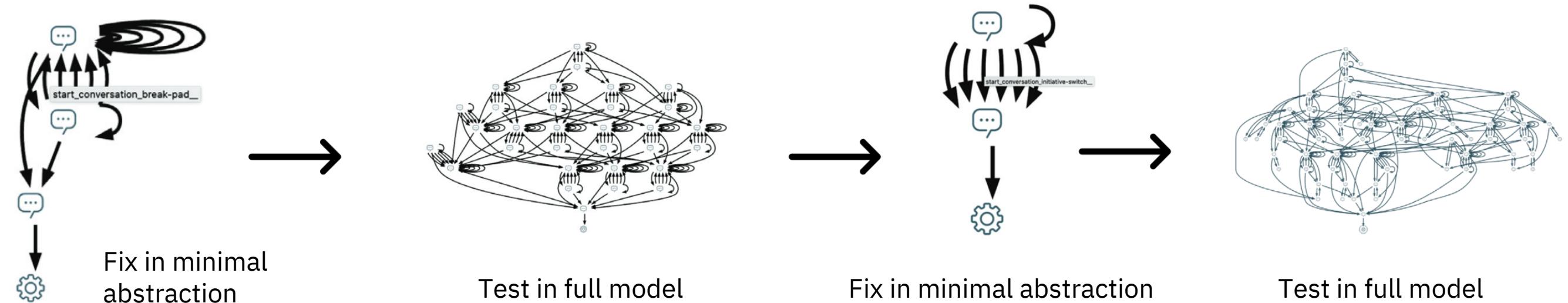
Designers must understand the imperative consequences of declarative design in order to trust the bots they build.

# Non-deterministic Planning as a design tool: Compose --> Debug



# Non-deterministic Planning as a design tool: Compose --> Debug

- Solvable Minimal Abstraction where a problem needs to be fixed.
- Maximal Abstraction where the problem persists.
- Unachievable Subgoal to highlight the problem.



Demo

# Interpretability for the End User

An online plan is a sequence.

Plan for most optimistic outcome and re-plan if that doesn't materialize at runtime.

```
Step 0: auth_agent ( loan_agent )
Step 1: slot_fill_agent ( loan_amount )
Step 2: slot_fill_agent ( account_number )
Step 3: slot_fill_agent ( credit_score )
Step 4: slot_fill_agent ( income )
Step 5: slot_fill_agent ( valid_id )
Step 6: loan_agent ( account_number,
income, credit_score, valid_id,
loan_amount )
```

For  
the  
end  
user.

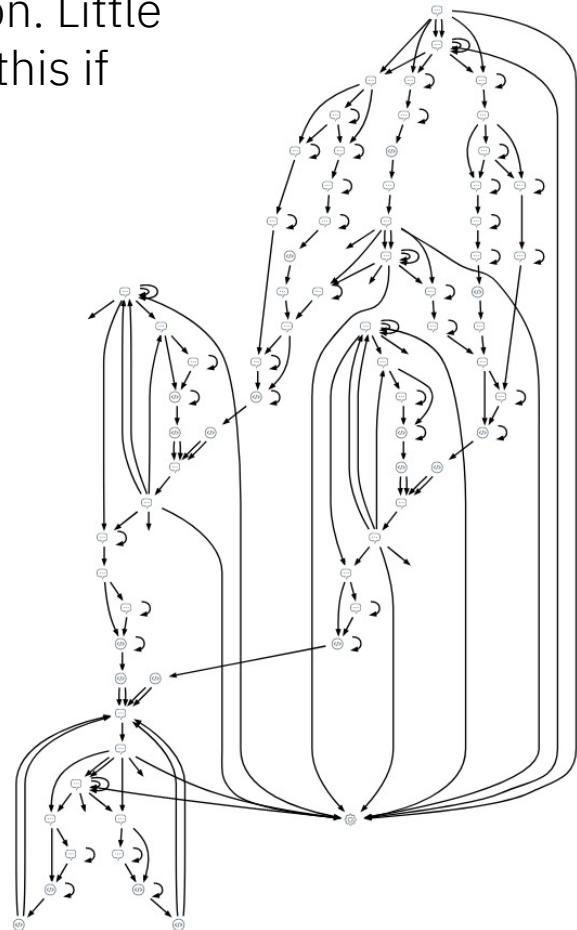
**Runtime:** Plan and re-plan. Easy to adapt to non-stationary domains.

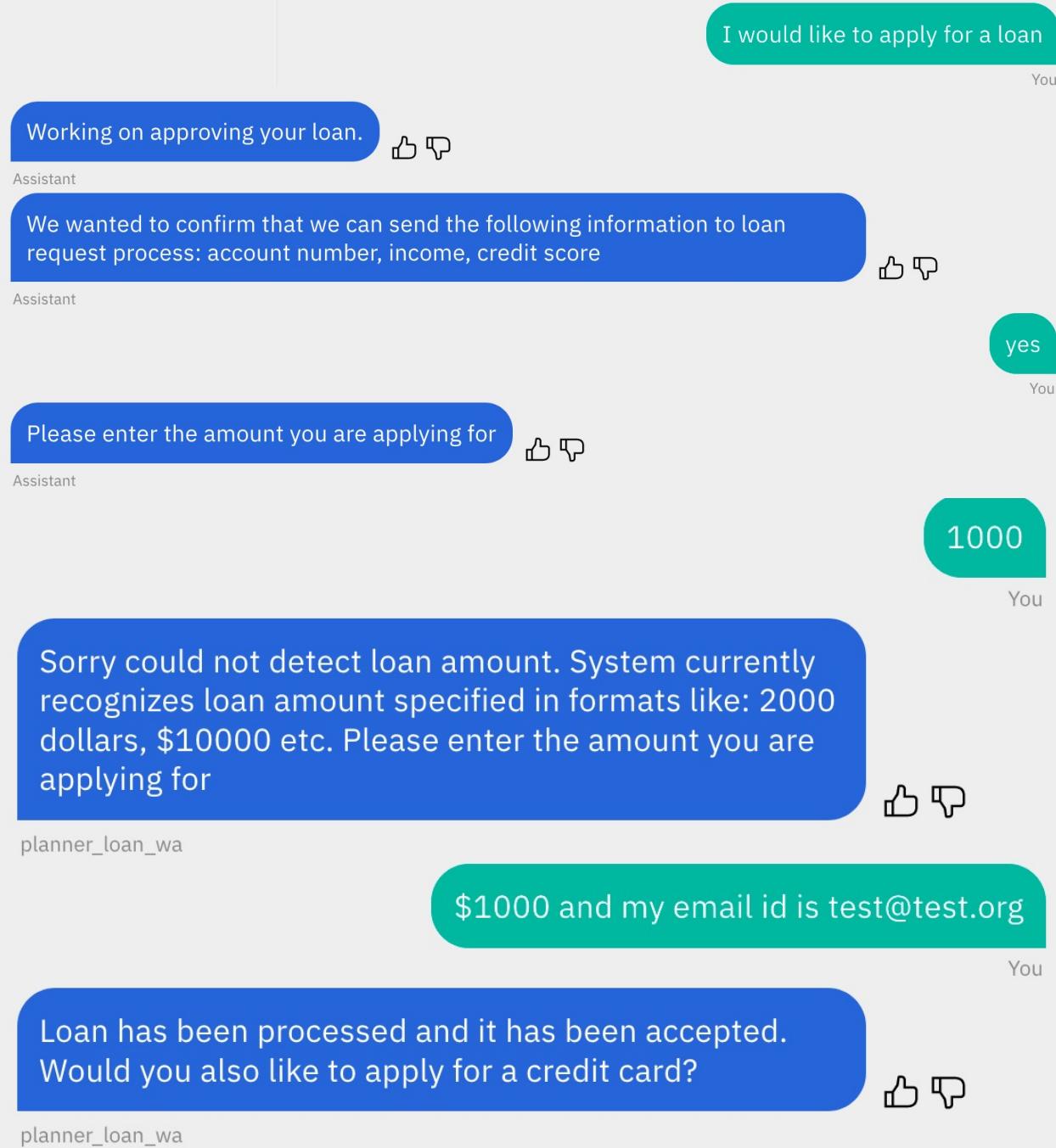
An offline “contingent” plan is a graph or tree.

Plan for all outcomes and determine which one happened at runtime.

**Runtime:** Outcome determination. Little to no reasoning. No point doing this if domain is not stationary.

**Bonus:** You can see what it is you have made.





### Planner attempt – 1

- Step 0: slot\_fill\_skill ( loan\_processed )
- Plan failed.

### Planner attempt – 2

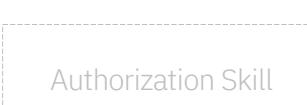
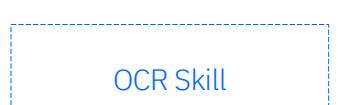
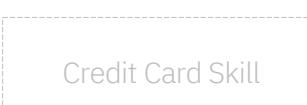
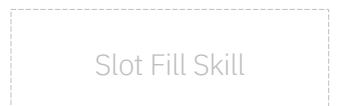
- Step 0: auth\_skill ( loan\_skill )
- Step 1: slot\_fill\_skill ( loan\_amount )
- Step 0: slot\_fill\_skill ( loan\_amount )
- Step 1: slot\_fill\_skill ( account\_number )
- Step 2: slot\_fill\_skill ( credit\_score )
- Step 3: slot\_fill\_skill ( income )
- Step 4: slot\_fill\_skill ( valid\_id )
- Step 5: loan\_skill ( account\_number, income, credit\_score, valid\_id, loan\_amount )

Unexpected state: re-plan.

### Planner attempt – 3

- Step 0: dbq\_skill ( email\_id, account\_number, income, credit\_score, license\_screenshot )
- Step 1: ocr\_skill ( license\_screenshot, valid\_id, ocr\_skill )
- Step 2: loan\_skill ( account\_number, income, credit\_score, valid\_id, loan\_amount )

Success.



# Transparency of Aggregated Assistants Explanatory Dialogue

Please give me a summary of what you did  
You  
I first collected information about valid id, income, credit score, account number and then I collected information about loan amount and then finally I collected information about loan status.  
planner\_loan\_wa

how did you get my credit score  
You  
Information collected from internal database query service. Which used the following information: email id.  
planner\_loan\_wa

How did you get my email id  
You  
Customer provided the value.  
planner\_loan\_wa

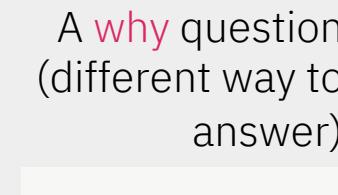
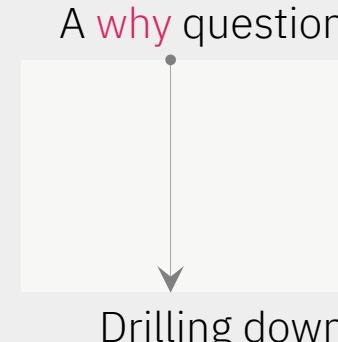
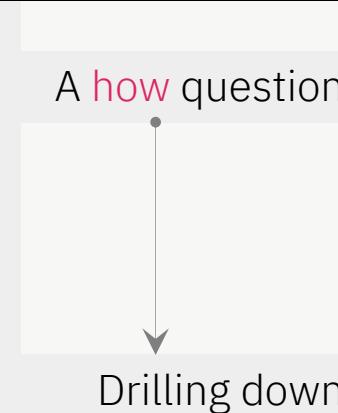
• • •

why did you collect my email id  
You  
The information was needed for internal database query service. Which generates the following required information: income, credit score, account number, license screenshot.  
planner\_loan\_wa

why did you need credit score  
You  
The information was needed for loan request process. Which generates the following required information: loan status.  
planner\_loan\_wa

OR

why did you collect my email id  
You  
That information was used by internal database query service. The rest of the sequence was as follows: internal database query service->text scanning process->loan request process  
planner\_loan\_wa



Landmark and causal link analysis for explanations.

○ **How** questions: Explain the “source” or provenance of data.

○ **Why** questions: Explain the “sink” of data. This can be done in two ways:

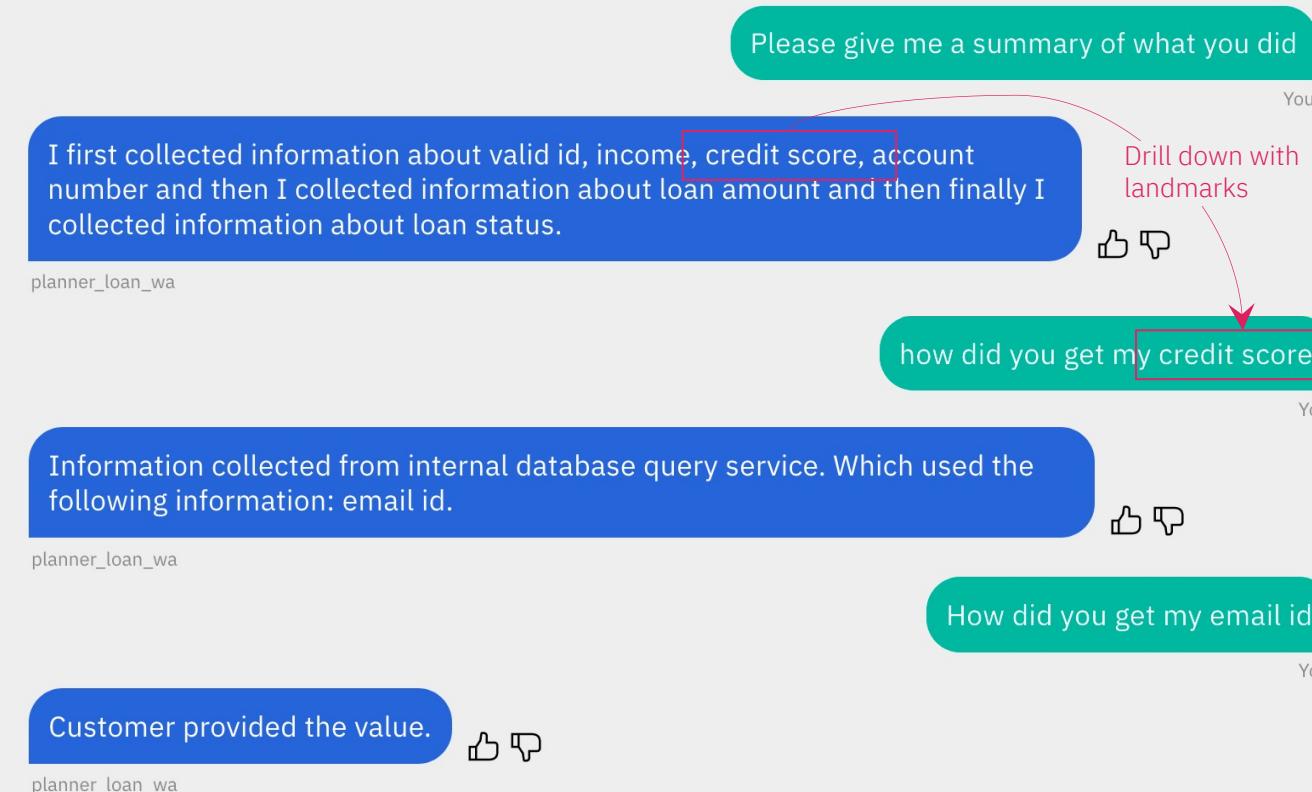
- Explain where the data was required.
- Explain what was done with the data.

# Transparency of Aggregated Assistants Explanatory Dialogue

Zoomed

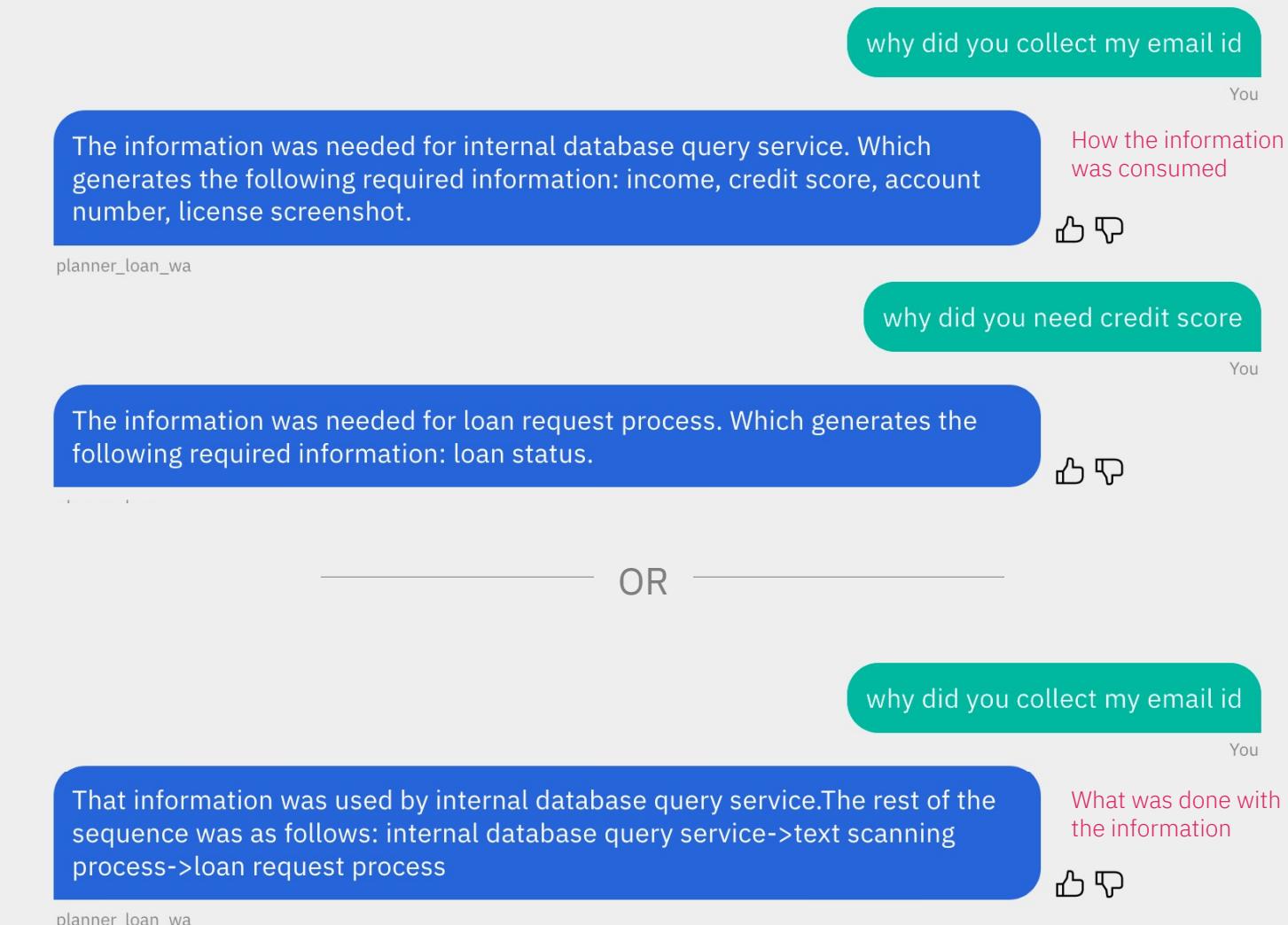
A *how* question

Summarize using state landmarks then drill down using the causal chain to achieve them.



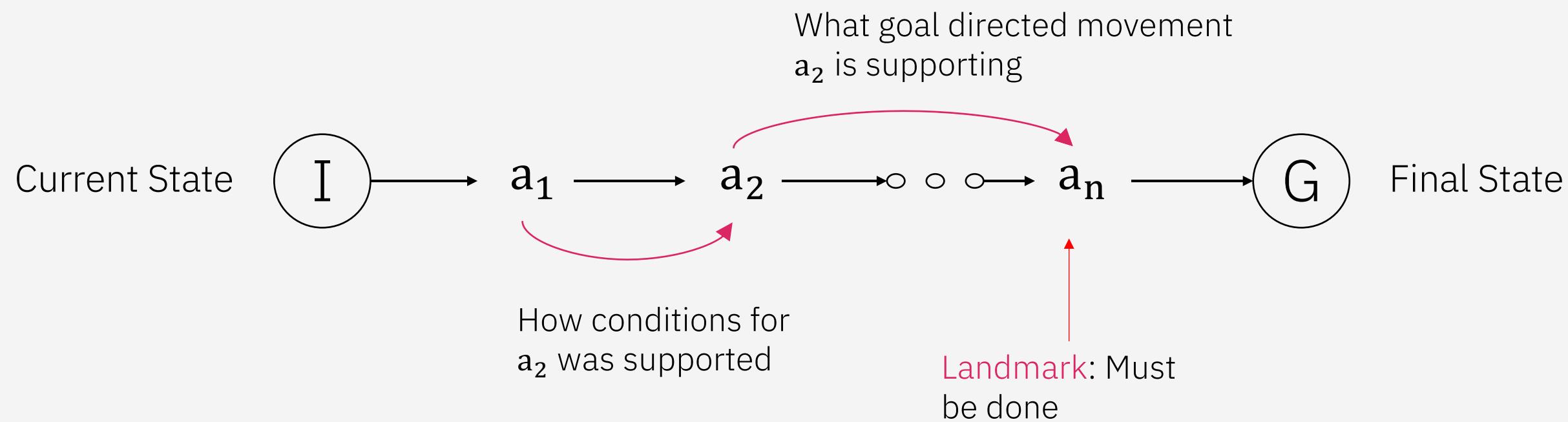
A *why* question

Answer with either how the outputs of the event was consumed recursively OR what was done based on that event.



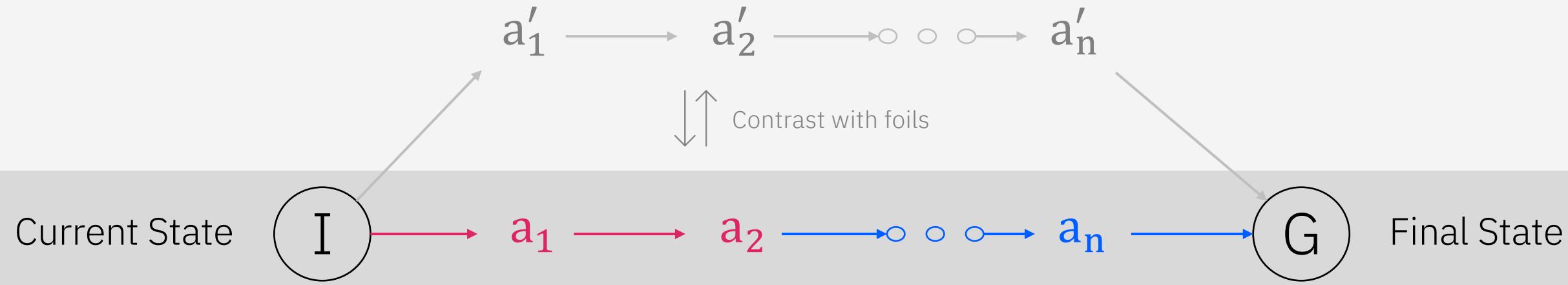
# Causal Links and Landmarks

for inference reconciliation



Foil: Why not this instead

---



Why it is  
being done

---

How it has  
been done

What has  
been done

# Interpretability Rewind

An online plan is a sequence.

Plan for most optimistic outcome and re-plan if that doesn't materialize at runtime.

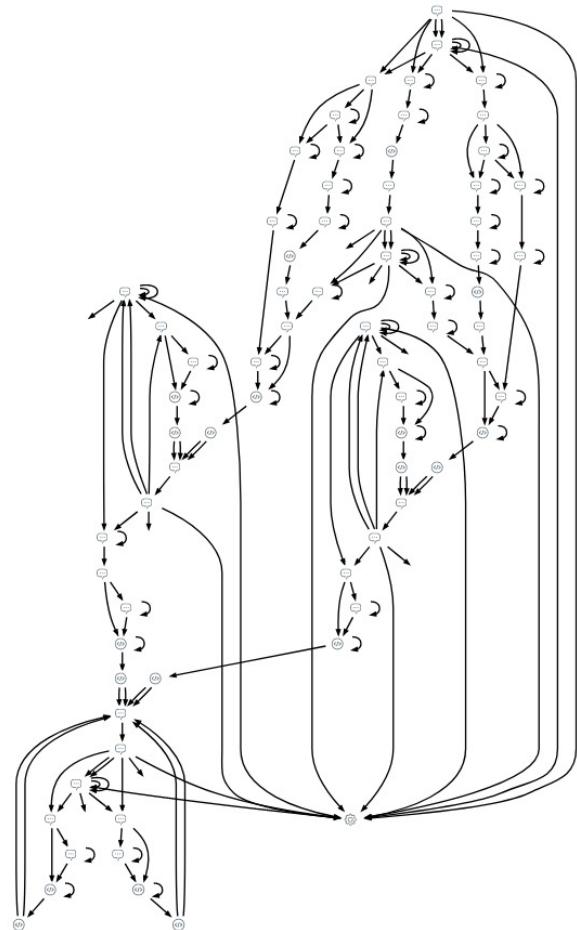
```
Step 0: auth_agent ( loan_agent )
Step 1: slot_fill_agent ( loan_amount )
Step 2: slot_fill_agent ( account_number )
Step 3: slot_fill_agent ( credit_score )
Step 4: slot_fill_agent ( income )
Step 5: slot_fill_agent ( valid_id )
Step 6: loan_agent ( account_number,
income, credit_score, valid_id,
loan_amount )
```

**Causal links and landmarks:** To establish provenance of data along a process.

An offline “contingent” plan is a graph or tree.

Plan for all outcomes and determine which one happened at runtime.

**Domain Abstractions:** In conjunction with causal links and landmarks, to highlight potential issues.



# References

[A Unified Conversational Assistant Framework for Business Process Automation.](#) Yara Rizk, Abhisekh Bhandwalder, Scott Boag, Tathagata Chakraborti, Vatche Isahagian, Yasaman Khazaeni, Falk Pollock, and Merve Unuvar. AAAI 2020 Workshop on Intelligent Process Automation.

[D3WA+ -- A Case Study of XAIP in a Model Acquisition Task for Dialogue Planning.](#) Sarath Sreedharan, Tathagata Chakraborti, Christian Muise, Yasaman Khazaeni, and Subbarao Kambhampati. ICAPS 2020.

[Planning for Goal-Oriented Dialogue Systems.](#) Christian Muise, Tathagata Chakraborti, Shubham Agarwal, Ondrej Bajgar, Arunima Chaudhary, Luis A. Lastras-Montano, Josef Ondrej, Miroslav Vodolan, and Charlie Wiecha. Technical Report (2019).

[D3BA: A Tool for Optimizing Business Processes Using Non-Deterministic Planning.](#) Tathagata Chakraborti, Shubham Agarwal, Yasaman Khazaeni, Yara Rizk, and Vatche Isahagian. BPM 2020 Workshop on AI4BPM.

[From Robotic Process Automation to Intelligent Process Automation: Emerging Trends.](#) Tathagata Chakraborti, Vatche Ishakian, Rania Khalaf, Yasaman Khazaeni, Vinod Vinod Muthusamy, Yara Rizk, and Merve Unuvar. BPM 2020 RPA Forum.

[Explainable Composition of Aggregate Assistants.](#) Sarath Sreedharan, Tathagata Chakraborti, Yara Rizk, and Yasaman Khazaeni. ICAPS 2020 Workshop on Explainable AI Planning.

In collaboration with [Yara Rizk](#) · [Sarah Sreedharan](#) · [Yasaman Khazaeni](#) · [Christian Muise](#) · [Shubham Agarwal](#) · [Scott Boag](#) · [Vatche Ishakian](#) · [Dario Silva Moran](#) · [Rania Khalaf](#) · [Vinod Muthusamy](#) · [Merve Unuvar](#) · [Luis Lastras](#) · [Ondrej Bajgar](#) · [Arunima Chaudhary](#) · [Abhisekh Bhandwalder](#) · [Falk Pollock](#) · [Josef Ondrej](#) · [Miroslav Vodolan](#) · [Charlie Wiecha](#) · [Subbarao Kambhampati](#)



1. New trends in Business Process Management
  2. Basics of Automated Planning
  3. Automated Planning along the BPM Life-cycle
    - Declarative modeling
    - Trace alignment against declarative process models
    - Interpretability
- ## 4. Concluding Remarks



IBM

# Off-the-Shelf Tools for Planning

Wanna build your own system?

Check out these amazing planning softwares build by the ICAPS community to get started. 😊

[Fast Downward](#) Planning System

[Tarski](#) An AI Planning Modeling Framework

[ROSPlan | ROS2](#) Planning and Robotics

[VAL](#) The Plan Validation System

[OPTIC | KCL Planners](#) with time and preferences

[PRP Planner](#) Non-deterministic planning

[Fast Forward](#) Family of satisficing planners

[IBM TOP-K Planners](#) Diverse and Top-Quality planning

[Pyperplan](#) A lightweight STRIPS planner in Python

[LAPKT](#) A lightweight automated planning toolkit

[Planutils](#) A linux-based planning environment

[Planning.Domains](#) Planning on the web

[VS Code | Sublime | Atom](#) PDDL Plugins

Don't see something here? Contribute to the Planning [GitHub](#), add to [Planning.Wiki](#), or send us an [email](#).

Automated Planning

PDDL tips and tricks!

- Link: <https://icaps21.icaps-conference.org/demos> right hand panel



- Planning models are **all general** in the sense that they are **not bound** to specific problems or domains.
  - **Scale-up** both in terms of 1) size of specification to complexity of process and 2) re-useability of compilations across different domains of interest.
- This **generality** is coupled with the notion of **intelligence** which requires the ability to deal with new problems.
- Planning models are **inherently interpretable** and thus aid human-in-the-loop authoring and interfacing to business processes.
- The price for generality is **computational**:
  - Planning over models represented in compact form is **intractable** in the worst case, yet currently large problems can be solved **very quickly**.