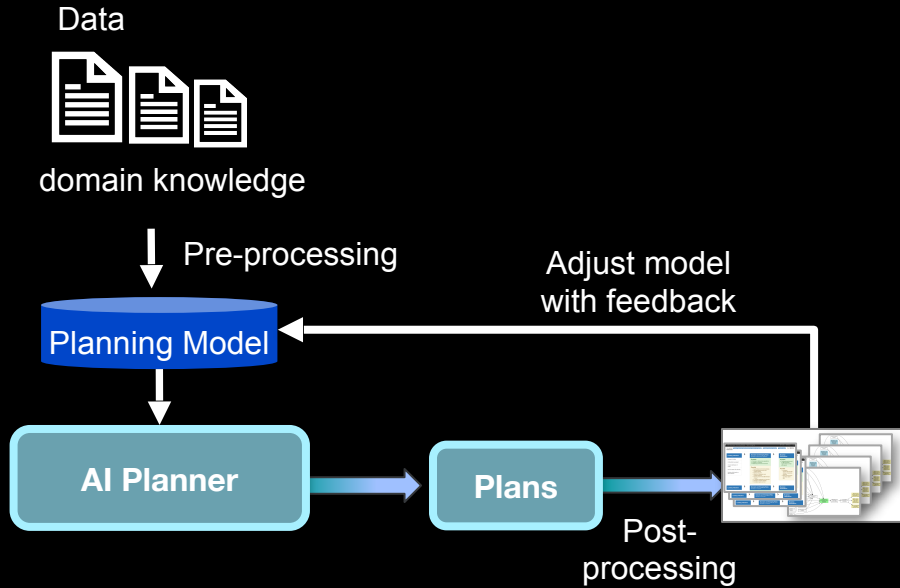


# Modeling



1. Theory

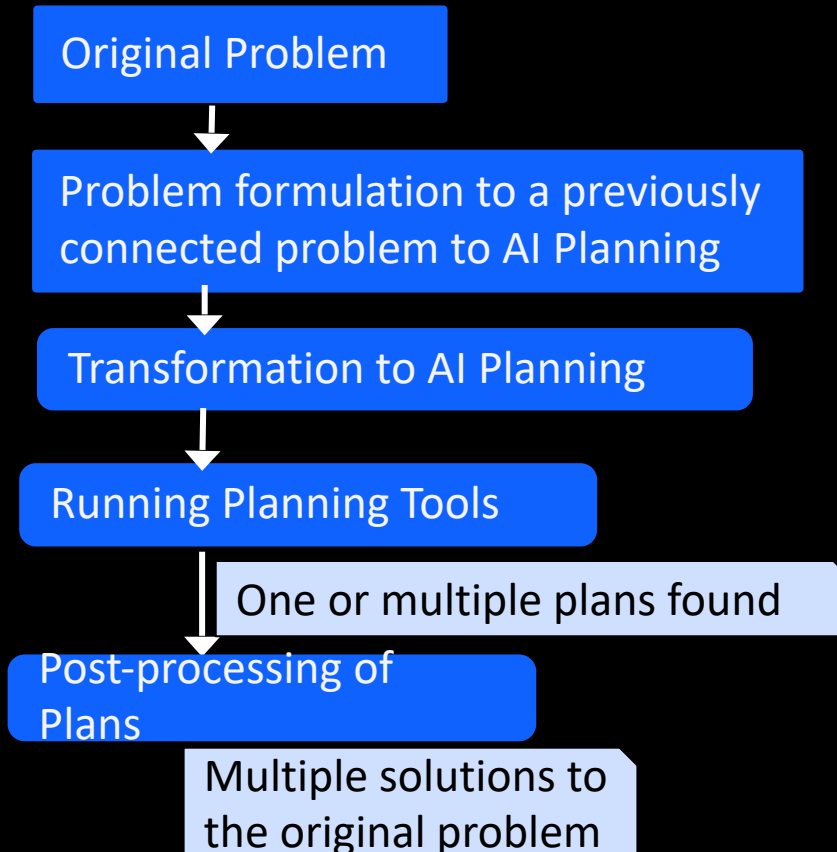
2. Modeling ←

3. Practice

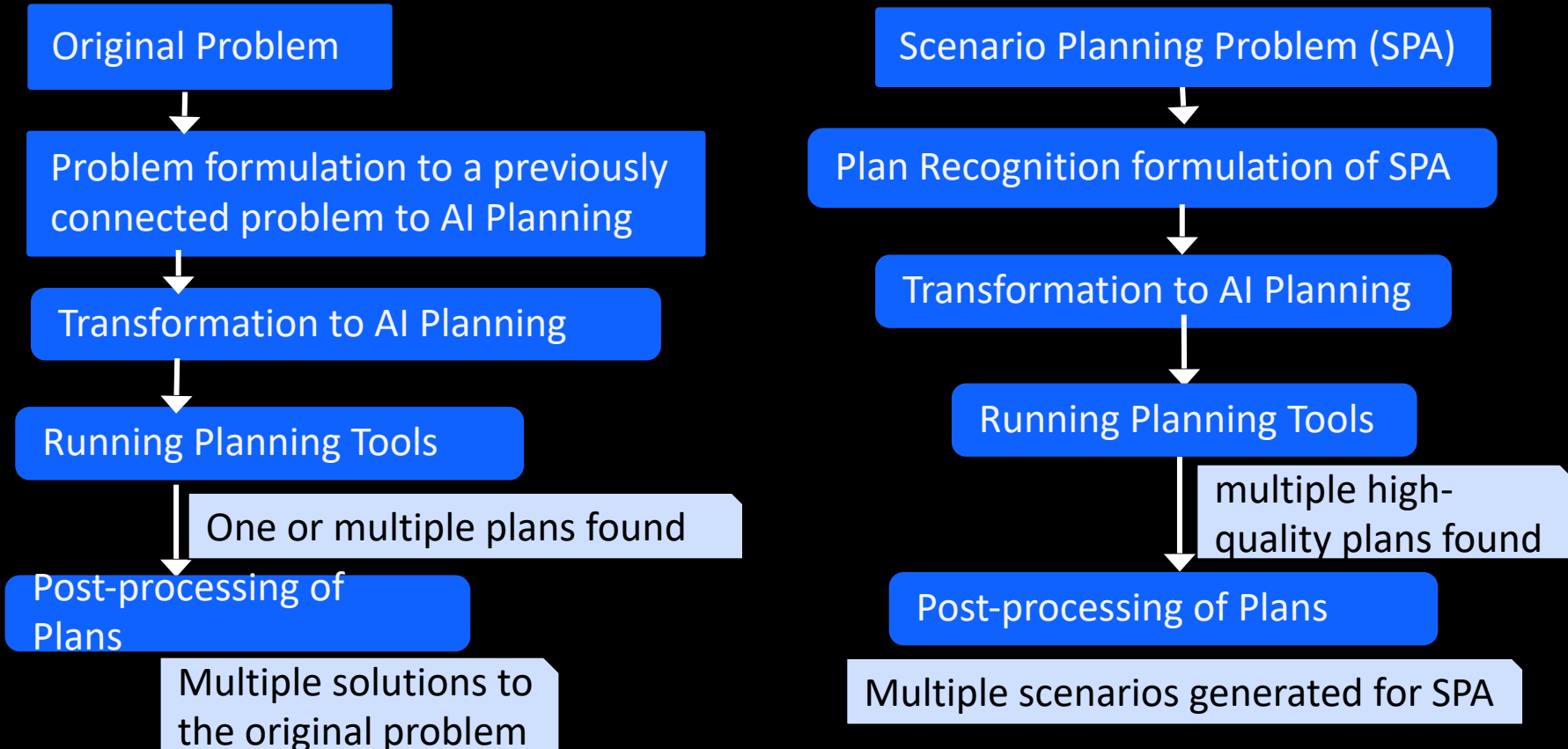
# Modeling Challenges

1. Often need to transform the original problem to a problem with known/easier to compute solutions
2. Access to domain experts familiar with input to planners (e.g., PDDL) is rare.
3. Any form of knowledge is practically guaranteed to be incomplete and often inconsistent. Even human validation can be ambiguous
4. Current state-of-art learning approaches may not scale and may have assumptions that may not hold for the practical setting. Further, the learned knowledge may be not consumable

# Relationship to Planning



# Relationship to Planning





# Established relationships to AI Planning

- Finding excuses (Göbelbecker et al. 2010)
- Diagnosis (Sohrabi et al., 2010)
- Explanation generation (Sohrabi et al., 2011)
- Plan Recognition (Ramirez & Geffner IJCAI'09, AAAI'10, Sohrabi et al., IJCAI'16)

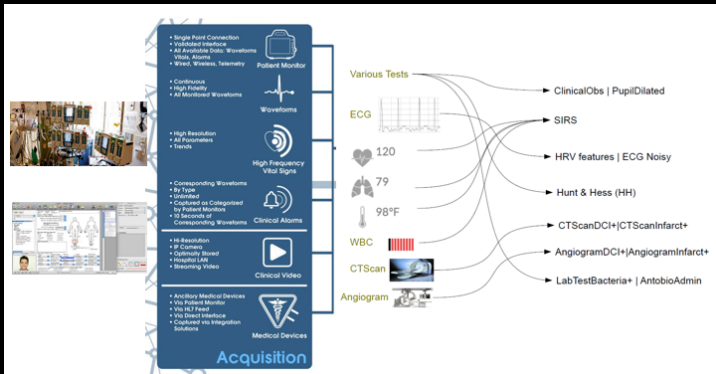
Which in turn helped with:

- Hypothesis Generation and Exploration (Sohrabi et al., 2013, 2015)
- Future State Projection (Sohrabi et al., 2017)
- Multi-agent Plan Recognition (Shvo et al., 2018)
- Scenario Planning (Sohrabi et al., 2018)

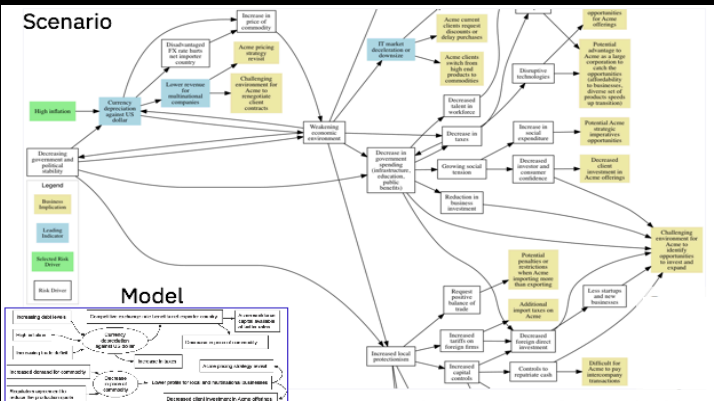
# Additional Compilations

- Compiling away soft goals (Keyder and Geffner, 2009)
  - Use any classical planner to obtain a solution for a planning problem with soft goals (i.e., simple preferences)
  
- Translate HTN Problems into classical planning problems (Alford et al., 2016)
  - Use classical planners (heuristic search) to find solutions for an HTN planning problem.

### Examples



[Automated large-scale data analysis, ICAPS 2015]



7 [Scenario planning for enterprise risk management, AAI 2018]

[D3WA+: A Case Study of XAIP in a Model Acquisition Task, ICAPS 2020]



[Exploring Context-Free Languages via Planning: The Case for Automating Machine Learning, ICAPS 2020]

# AutoAI: Automating ML Pipeline Generation



## Problem

Space of possible pipelines is huge

Humans can explore only a tiny portion of it

Humans are biased towards the pipelines they already are familiar with

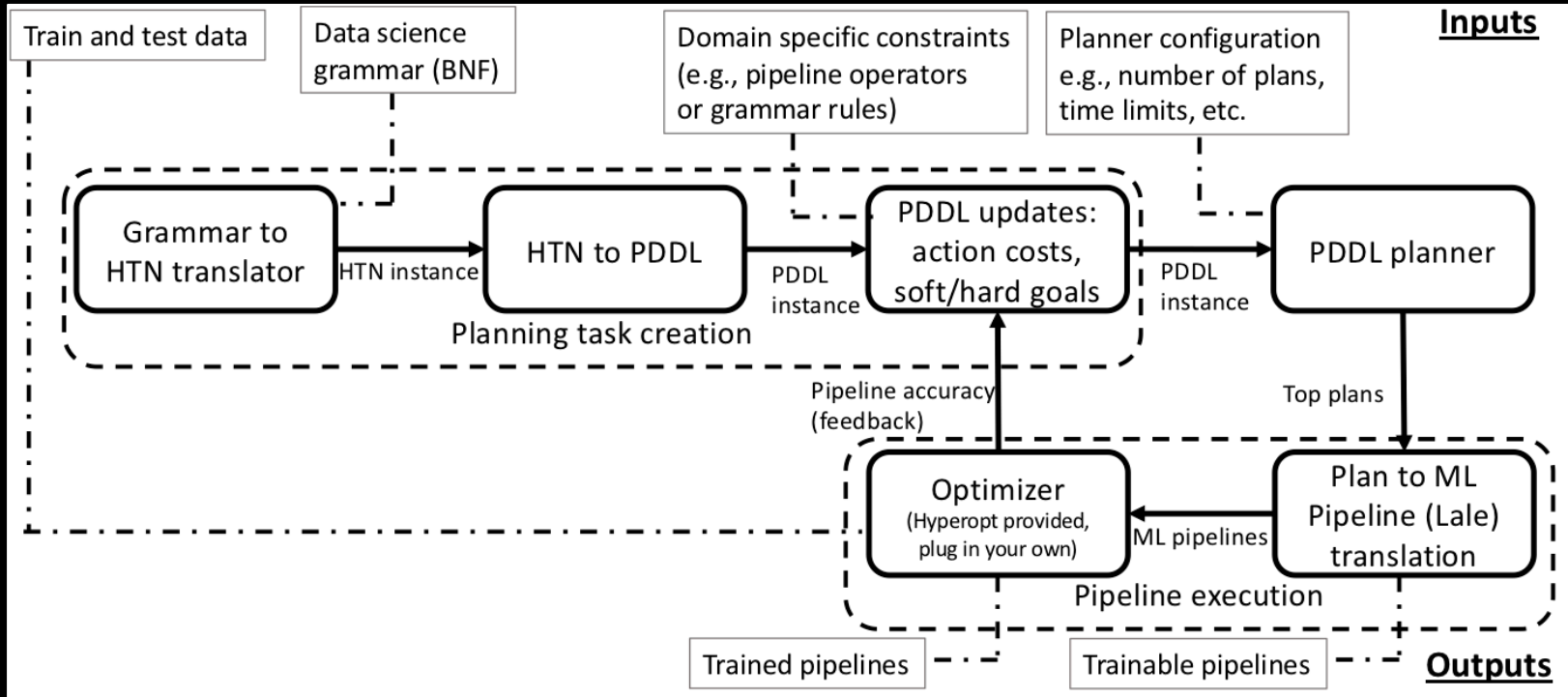
## Benefits

- Generation of pipelines of high accuracy automatically
- Reduces the need for human data scientists

## Solution

- Use regular grammars to define the possible pipeline compositions
- Translate regular grammars to HTN planning model, and then to classical planning model
- Enrich planning model with user defined constraints
- Use AI Planners to solve the planning model and translate plans to pipelines

# Automating ML Pipeline Generation: Solution



# User Guided Exploration: Choosing Preferred Symbols

- **Non-terminal symbols — rules**
- **Terminal symbols — specific algorithms and hyper-param values**
  
- **Translated to soft goals**
- **Soft goals compiled away (Keyder & Geffner 2009)**

num\_pipeli... 29

Search con...

- GradientBoostingClassifier
- KNeighborsClassifier
- KeepNonNumbers
- KeepNumbers
- LogisticRegression
- MLPClassifier
- MinMaxScaler()
- NoOp()
- Normalizer()
- Nystroem
- OneHotEncoder
- PCA
- PolynomialFeatures()
- QuadraticDiscriminantAnalysis()
- RandomForestClassifier
- RobustScaler()
- SimpleImputer
- StandardScaler()
- activation
- criterion
- dag

Get pipelines

# Representing the Knowledge (Example: PDDL)

```
(define (problem mixed)
  (:domain miconic)
  (:objects p0 p1 p2 p3 f0 f1 f2 f3 f4 f5 f6 f7)

  (:init
   (passenger p0)(passenger p1)(passenger p2)(passenger p3)
   (floor f0)(floor f1)(floor f2)(floor f3)(floor f4)
   (floor f5)(floor f6)(floor f7)

   (above f0 f1)(above f0 f2)(above f0 f3)(above f0 f4)
   (above f0 f5)(above f0 f6)(above f0 f7)(above f1 f2)
   (above f1 f3)(above f1 f4)(above f1 f5)(above f1 f6)
   (above f1 f7)(above f2 f3)(above f2 f4)(above f2 f5)
   (above f2 f6)(above f2 f7)(above f3 f4)(above f3 f5)
   (above f3 f6)(above f3 f7)(above f4 f5)(above f4 f6)
   (above f4 f7)(above f5 f6)(above f5 f7)(above f6 f7)

   (origin p0 f0)(destin p0 f5)(origin p1 f7)(destin p1 f4)
   (origin p2 f0)(destin p2 f7)(origin p3 f1)(destin p3 f6)

   (lift-at f0))

  (:goal (and
   (served p0)(served p1)(served p2)(served p3))))
```

```
(define (domain miconic)
  (:requirements :strips)

  (:predicates
   (origin ?person ?floor)
   (floor ?floor)
   (passenger ?passenger)
   (destin ?person ?floor)
   (above ?floor1 ?floor2)
   (boarded ?person)
   (served ?person)
   (lift-at ?floor))

  (:action board
   :parameters (?f ?p)
   :precondition (and (floor ?f) (passenger ?p)(lift-at ?f) (origin ?p ?f))
   :effect (boarded ?p))

  (:action depart
   :parameters (?f ?p)
   :precondition (and (floor ?f) (passenger ?p) (lift-at ?f) (destin ?p ?f)
    (boarded ?p))
   :effect (and (not (boarded ?p))
    (served ?p)))

  (:action up
   :parameters (?f1 ?f2)
   :precondition (and (floor ?f1) (floor ?f2) (lift-at ?f1) (above ?f1 ?f2))
   :effect (and (lift-at ?f2) (not (lift-at ?f1))))

  (:action down
   :parameters (?f1 ?f2)
   :precondition (and (floor ?f1) (floor ?f2) (lift-at ?f1) (above ?f2 ?f1))
   :effect (and (lift-at ?f2) (not (lift-at ?f1))))
```

(up fo fi)

(up fi f7)

(board f7 p1)

(down f7 f4)

(depart f4 p1)

(down f4 fo)

(board fo po)

(up fo f5)

(depart f5 po)

(up f5 f6)

(down f6 fo)

(board fo p2)

(up fo f7)

(depart f7 p2)

(down f7 fi)

(board fi p3)

(up fi f6)

(depart f6 p3)



# Modeling Examples

```

components.fpl 11
/**
 * @type "pig"
 * @title "Top K By Count"
 * @tags o TopKByCount
 */
= component TopSummary(input i; output o) : SummaryK {
  param SK;
}
/**
 * @type "pig"
 * @title "Order By Desc"
 * @tags o OrderByDesc
 */
= component OrderByDesc(input i; output o) : OrderByDesc {
  param DESC;
}
/**
 * @type "jaql"
 * @title "Count Flows by Destination Host"
 * @tags o FlowCountByDestinationHost
 */
= component CountFlowsByDestinationHost(input i; output o) : CountFlows {
  param DEST_HOST;
}
/**
 * @type "jaql"
 * @title "Live DNS Data"
 * @tags o LiveDNS
 */
= component LiveDNS(output o) {
  param SCOPES;
}
/**
 * @type "main"
 * @title "Main (top-level) composite for this pattern"
 * @tags
 */
= composite MyMain(output final) {
  graph
  stream data = Source() { param databaseName : UserParam("DBFS Source");
  stream filterData = FilterByDestinationHost();
  stream aggregate = Summarize();
  stream final = SummarySink();
}

```

Annotations in the image:

- PIG component
- JAQL component
- SPL component
- Platform specific code for the component
- Semantic annotations (tags) on the component ports
- Pattern (composite) with variability points
- Cascade is used by data scientists/analytic

```

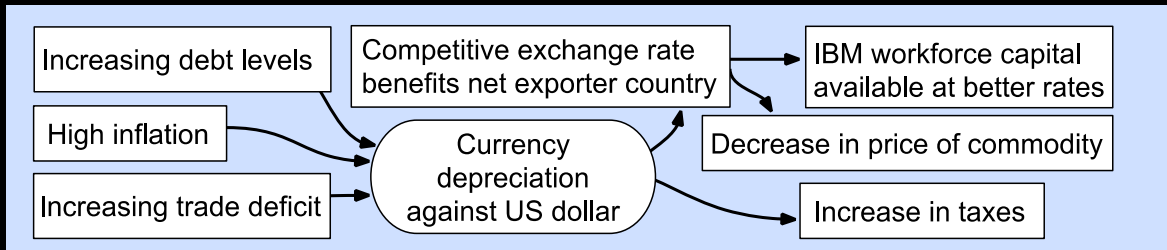
1 default-class bad_state
2 Lifecycle = (start {<good_state>} -> ADMITTED)
3 ADMITTED = (admitted {OADMITTED} -> LOWRISK | admitted -> HIGHRISK)
4 LOWRISK = (lowrisk {<good_state>} OSIRS0 OHH1 -> HIGHRISK |
5 lowrisk -> DISCHARGE)
6 HIGHRISK = (highrisk {OSIRS2 OHH3 OHH4 OHRVL OCTSCANDCINEGATIVE}
7 -> PRECOMPLICATION | highrisk -> LOWRISK)
8 PRECOMPLICATION = (precomp {OSIRS2 OHH3 OHRVL} -> SUSPECTEDDCI |
9 precomp -> SUSPECTEDINFECTION)
10 SUSPECTEDDCI = (suspecteddci {OSIMDCI OVISUALDCI} -> PREDCI)
11 PREDCI = (predci {OPREDDCI OVISUALDCI OSIMDCI} -> DCI)
12 DCI = (dci {OANGIOGRAMDCIPOSITIVE} -> HIGHRISK |
13 dci -> PRECOMPLICATION | dci {OFLAT} -> ICUDEATH)
14 INFECTION = (infection {OINFECTIONPOSITIVE} -> HIGHRISK |
15 infection {OANTIBIOTICSADMINISTERED} -> PRECOMPLICATION |
16 infection {OFLAT} -> ICUDEATH)
17 ICUDEATH = (icudeath {OFLAT})
18 DISCHARGE = (discharge {OPATIENTDISCHARGED} -> discharge)
19 starting start

```

Line 1:1: The state transition graph contains a cycle with states: dci.predci.suspecteddci.precomp.highrisk,lowrisk

Cascade (Ranganathan et al., 2009)

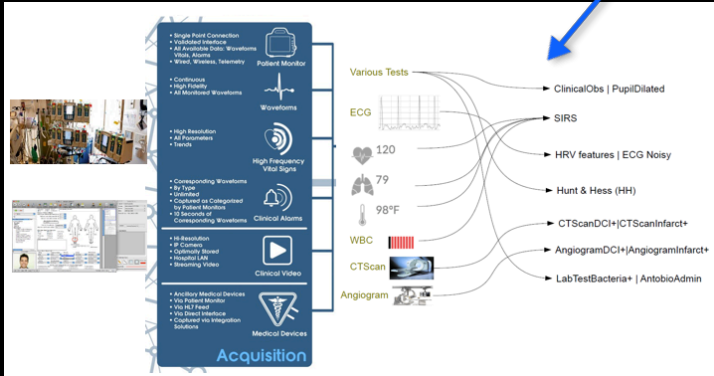
LTS++ (Sohrabi et al., 2015, 2020)



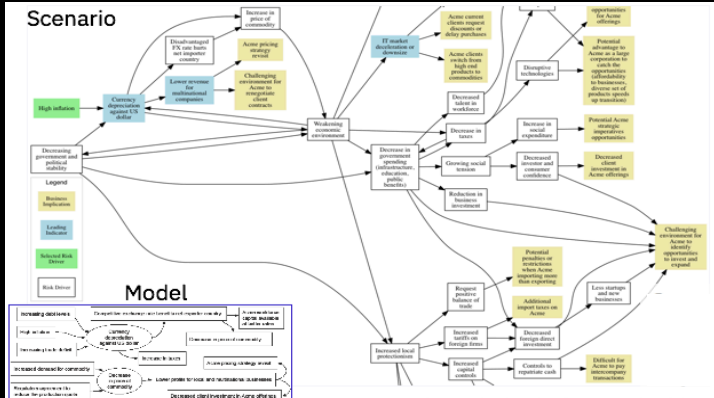
MindMaps (Sohrabi et al., 2018)



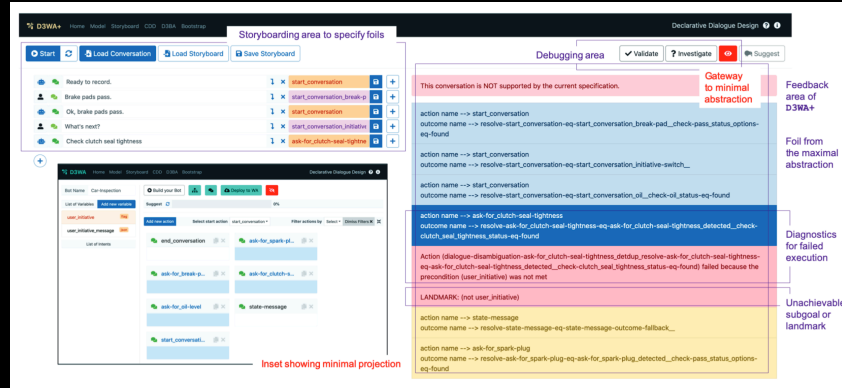
# Examples



[Automated large-scale data analysis, ICAPS 2015]



13 [Scenario planning for enterprise risk management, AAI 2018]

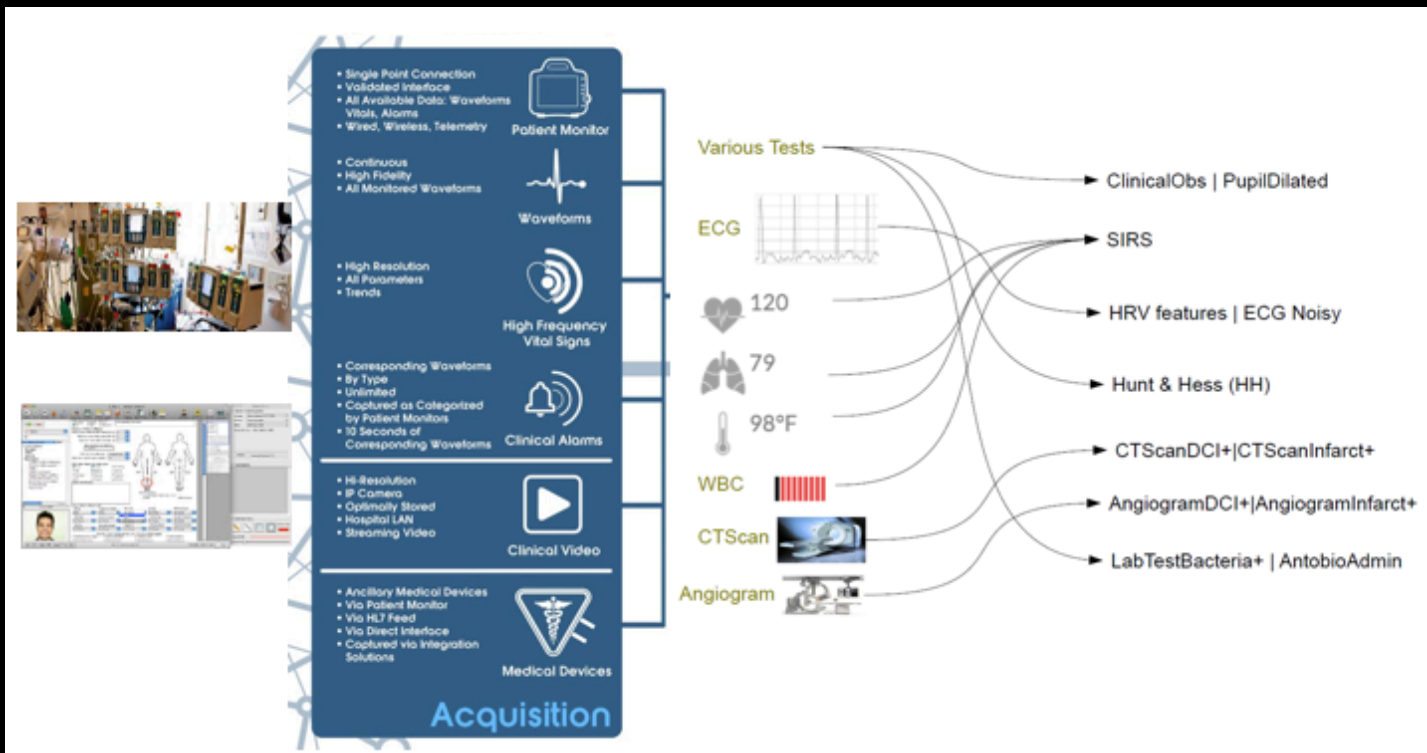


[D3WA+: A Case Study of XAIP in a Model Acquisition Task, ICAPS 2020]

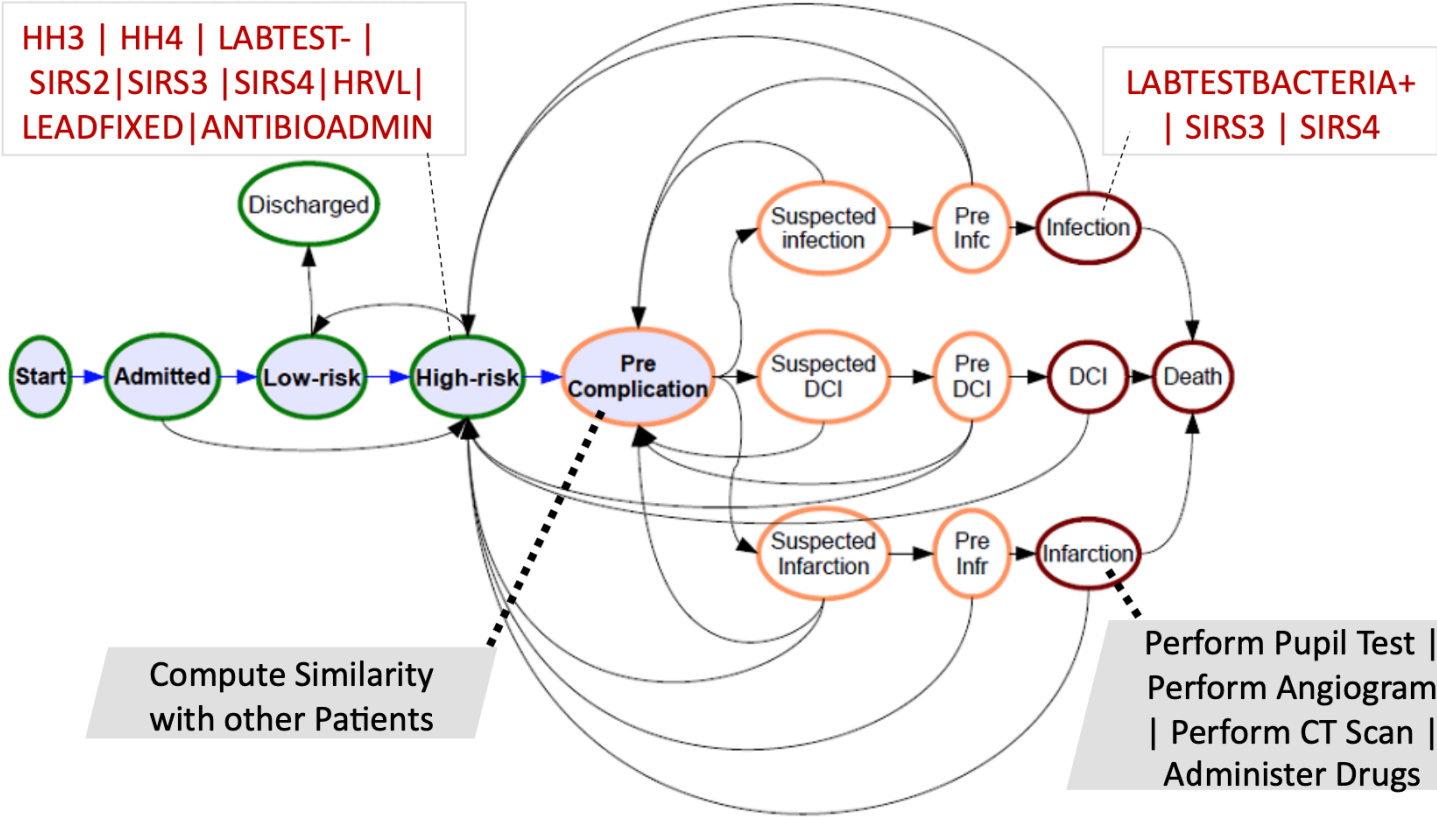


[Exploring Context-Free Languages via Planning: The Case for Automating Machine Learning, ICAPS 2020]

# Hypothesis Generation



# Patient Complication Detection



# LTS++

```
1 ⚠ default-class bad_state
2 LIFECYCLE = (start {<good_state>} -> UNADMITTED)
3 UNADMITTED = (unadmitted {<good_state>} -> LOWRISK | unadmitted -> HIGHRISK)
4 LOWRISK = (lowrisk {OSIRS0,OSIRS1,OHH1,OHH2,OAntibioticsAdministered} -> HIGHRISK | lowrisk -> DISCHARGE)
5 HIGHRISK = (highrisk {OSIRS2,OSIRS3,OSIRS4,OHH3,OHH4,OHRVL,OHRVMGoingDown,OLeadFixed,
6             OAntibioticsAdministered,OLabTestNegative} -> LOWRISK | highrisk -> INFARCTION |
7             highrisk -> INFECTION | highrisk -> PATIENTNOLEAD | highrisk -> DCI)
8 INFECTION = (infection {OHRVL,OLabTestBacteria,OSIRS3,OSIRS4} -> HIGHRISK |
9             infection -> ICUDEATH | infection -> PATIENTNOLEAD | infection -> LOWRISK)
10 INFARCTION = (infarction {OAngiogramInfarctionPositive,OCTScanInfarctionPositive,OPupilDilated} -> HIGHRISK |
11             infarction -> ICUDEATH | infarction -> PATIENTNOLEAD)
12 DCI = (dci {OAngiogramDCIPositive,OCTScanDCIPositive,OClinicalObsDCIPositive,OHRVL} -> HIGHRISK |
13        dci -> ICUDEATH | dci -> PATIENTNOLEAD)
14 ICUDEATH = (icudeath {OFlat})
15 PATIENTNOLEAD = (patientnolead {OHRVL,OECGNoisy} -> HIGHRISK)
16 DISCHARGE = (discharge {OPatientLeft} -> UNADMITTED)
17 starting start
18
19
```

⚠ Line 1:1: The state transition graph contains a cycle with states: discharge,dci,infection,patientnolead,infarction,highrisk,lowrisk,unadmitted

- Derived from an existing language, Labeled Transition System (LTS)
- Can associate an observation with a state, can specify class type

# Partial encoding of the sample example

```
(:action explain-observation
:parameters(?x - state ?obs1 - obs ?obs2 - obs ?cat - obs-type)
:precondition (and (is-next-obs ?obs1 ?obs2) (matches ?obs1 ?cat)
                  (explains ?x ?cat) (at-state ?x) (at-obs ?obs1))
:effect (and (not (at-obs ?obs1)) (at-obs ?obs2) (ready)
            (increase (total-cost) 0)))

(:action discard-observation
:parameters(?x - state ?obs1 - obs ?obs2 - obs)
:precondition (and (is-next-obs ?obs1 ?obs2) (at-state ?x) (at-obs ?obs1))
:effect (and (not (at-obs ?obs1)) (at-obs ?obs2) (ready)
            (increase (total-cost) 2000)))

(:action state-change
:parameters(?x - state ?y - state ?obs - obs)
:precondition (and (is-next-state ?x ?y) (at-obs ?obs) (at-state ?x) (ready))
:effect (and (not (at-state ?x)) (not (ready)) (entering-state ?y)
            (increase (total-cost) 0)))

(:action enter-state-good
:parameters(?y - state ?obs - obs)
:precondition (and (at-obs ?obs) (entering-state ?y) (good-state ?y))
:effect (and (at-state ?y) (not (entering-state ?y))
            (increase (total-cost) 1)))

(:action enter-state-bad
:parameters(?y - state ?obs - obs)
:precondition (and (at-obs ?obs) (entering-state ?y) (bad-state ?y))
:effect (and (at-state ?y) (not (entering-state ?y))
            (increase (total-cost) 10)))

(:action allow-unobserved
:parameters(?x - state ?obs - obs)
:precondition (and (at-obs ?obs) (at-state ?x))
:effect (and (ready) (increase (total-cost) 1100)))
```

# Partial encoding of the sample example

```
(:init
  (at-state admitted) (at-obs o_1) (ready)
```

```
(matches o_1 OHH1) (matches o_2 OSIRS0) (matches o_3 OSIRS2)
```

```
(explains lowrisk OSIRS0) (explains highrisk OSIRS2)
(explains precomp OSIRS2) (explains lowrisk OHH1)
(explains dci OANGIOGRAMDCIPOSITIVE)
(explains highrisk OHRVL) (explains precomp OHRVL)
```

```
(is-next-state admitted highrisk) (is-next-state admitted lowrisk)
(is-next-state lowrisk highrisk) (is-next-state highrisk lowrisk)
(is-next-state highrisk precomp) (is-next-state dci highrisk)
(is-next-state dci icudeath) (is-next-state dci precomp)
```

```
(bad-state dci) (bad-state highrisk) (good-state lowrisk)
```

```
(is-next-obs o_1 o_2) (is-next-obs o_2 o_3) (is-next-obs o_3 o_end))
```

```
(:goal (and (at-obs o_end) (ready)))
```

```
(:action explain-observation
:parameters(?x - state ?obs1 - obs ?obs2 - obs ?cat - obs-type)
:precondition (and (is-next-obs ?obs1 ?obs2) (matches ?obs1 ?cat)
                  (explains ?x ?cat) (at-state ?x) (at-obs ?obs1))
:effect (and (not (at-obs ?obs1)) (at-obs ?obs2) (ready)
            (increase (total-cost) 0)))
```

```
(:action discard-observation
:parameters(?x - state ?obs1 - obs ?obs2 - obs)
:precondition (and (is-next-obs ?obs1 ?obs2) (at-state ?x) (at-obs ?obs1))
              (bs ?obs1) (at-obs ?obs2) (ready)
              (se (total-cost) 2000)))
```

```
?y - state ?obs - obs)
next-state ?x ?y) (at-obs ?obs) (at-state ?x) (ready))
tate ?x)) (not (ready)) (entering-state ?y)
ease (total-cost) 0)))
```

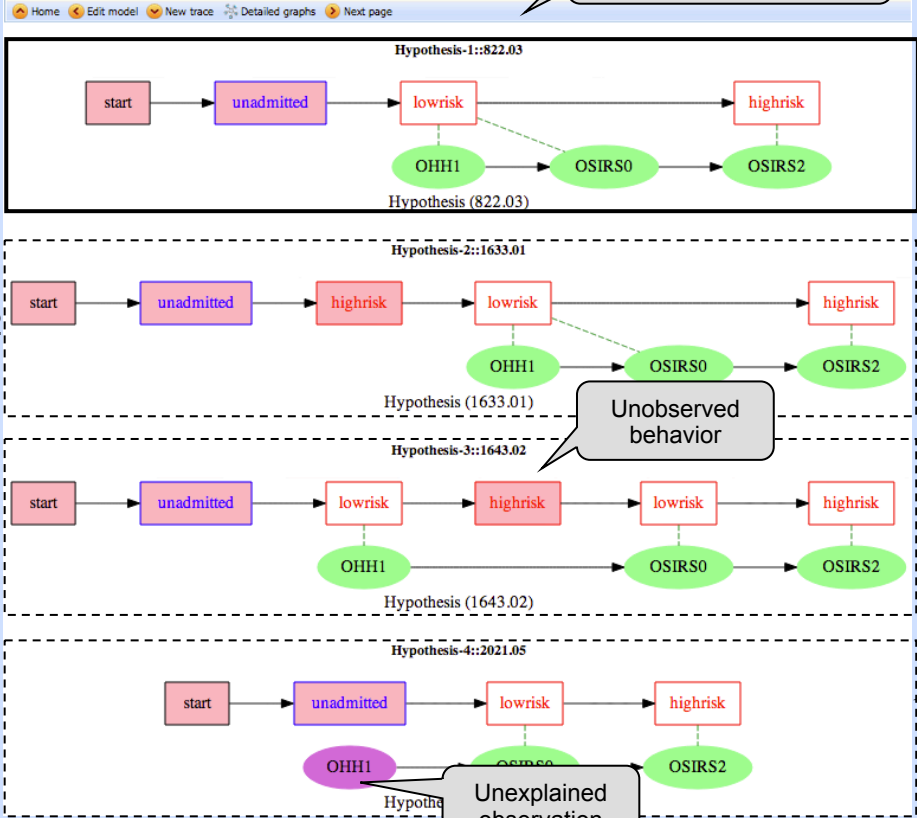
```
d
?obs - obs)
obs ?obs) (entering-state ?y) (good-state ?y))
?y) (not (entering-state ?y))
(total-cost) 1)))
```

```
?obs - obs)
obs ?obs) (entering-state ?y) (bad-state ?y))
?y) (not (entering-state ?y))
(total-cost) 10)))
```

```
(:action allow-unobserved
:parameters(?x - state ?obs - obs)
:precondition (and (at-obs ?obs) (at-state ?x))
:effect (and (ready) (increase (total-cost) 1100)))
```

# Generated hypotheses for the critical care application

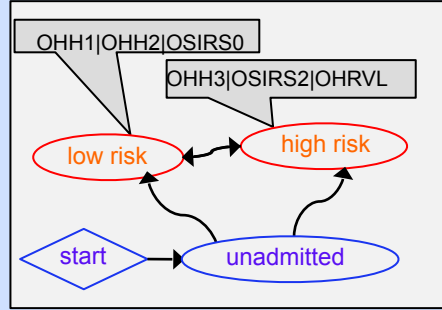
## Hypothesis Generation Results



Both OHH1 and SIRS0 are explained by lowrisk

Each hypothesis is shown as sequence of states matched to observed event sequence

### State Transition Diagram



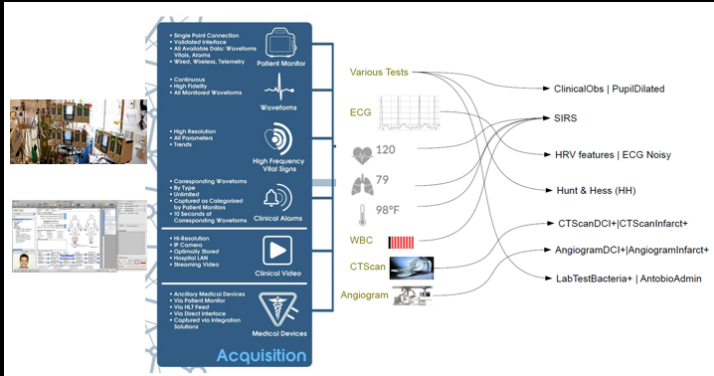
Unobserved behavior

Less plausible hypothesis

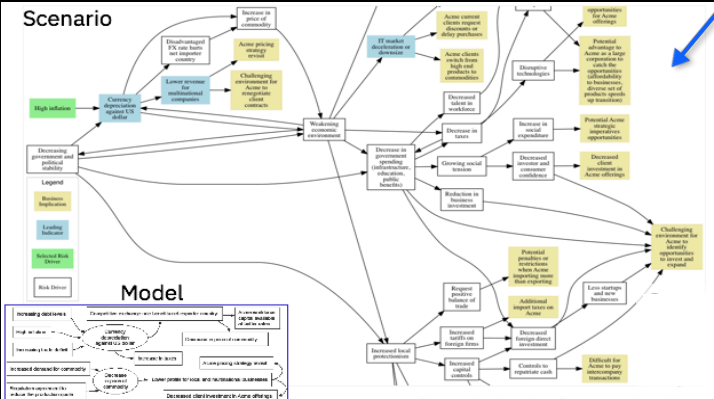
Unexplained observation



# Examples



[Automated large-scale data analysis, ICAPS 2015]



20 [Scenario planning for enterprise risk management, AAI 2018]

**Storyboarding area to specify foils**

- Start
- Load Conversation
- Save Storyboard

**Debugging area**

- Validate
- Investigate
- Suggest

**Feedback area of D3WA+**

This conversation is NOT supported by the current specification.

**Gateway to minimal abstraction**

action name -> start\_Conversation  
outcome name -> resolve-start\_Conversation-eq-start\_Conversation\_break-pad\_check-pass\_status\_options-eq-found

**Foil from the maximal abstraction**

action name -> start\_Conversation  
outcome name -> resolve-start\_Conversation-eq-start\_Conversation\_initiative\_switch-...

**Diagnostics for failed execution**

Action (dialogue-disambiguation-ask\_for\_clutch-seal-tightness\_detectup\_resolve-ask\_for\_clutch-seal-tightness-eq-ask\_for\_clutch-seal-tightness\_detected\_check-clutch\_seal\_tightness\_status-eq-found) failed because the precondition (user\_initiative) was not met.

**Unachievable subgoal or landmark**

LANDMARK: (not user\_initiative)

action name -> state-message  
outcome name -> resolve-state-message-eq-state-message-outcome-fallback-...

action name -> ask\_for\_spark-plug  
outcome name -> resolve-ask\_for\_spark-plug-eq-ask\_for\_spark-plug\_detected\_check-pass\_status\_options-eq-found

*Inset showing minimal projection*

[D3WA+: A Case Study of XAIP in a Model Acquisition Task, ICAPS 2020]



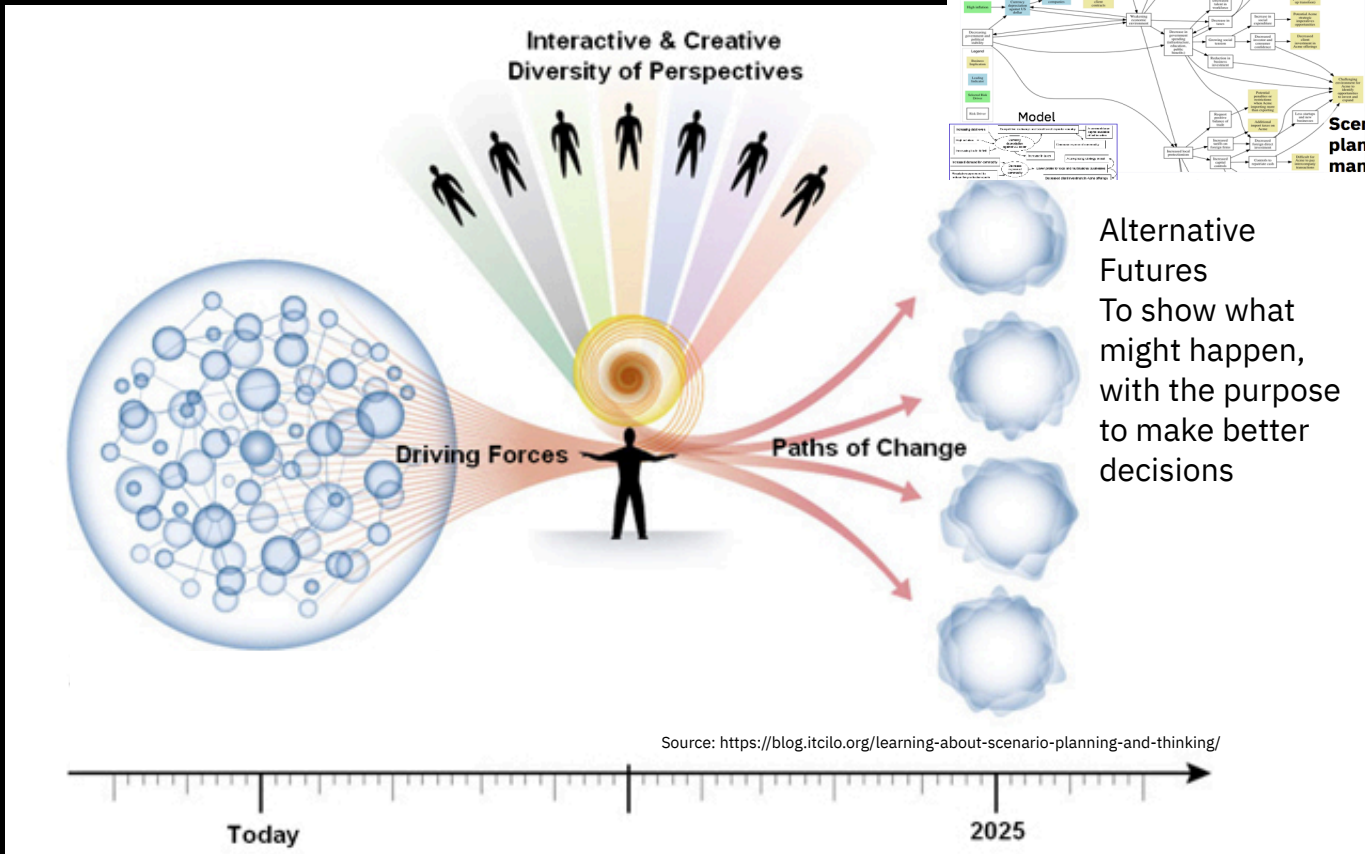
[Exploring Context-Free Languages via Planning: The Case for Automating Machine Learning, ICAPS 2020]



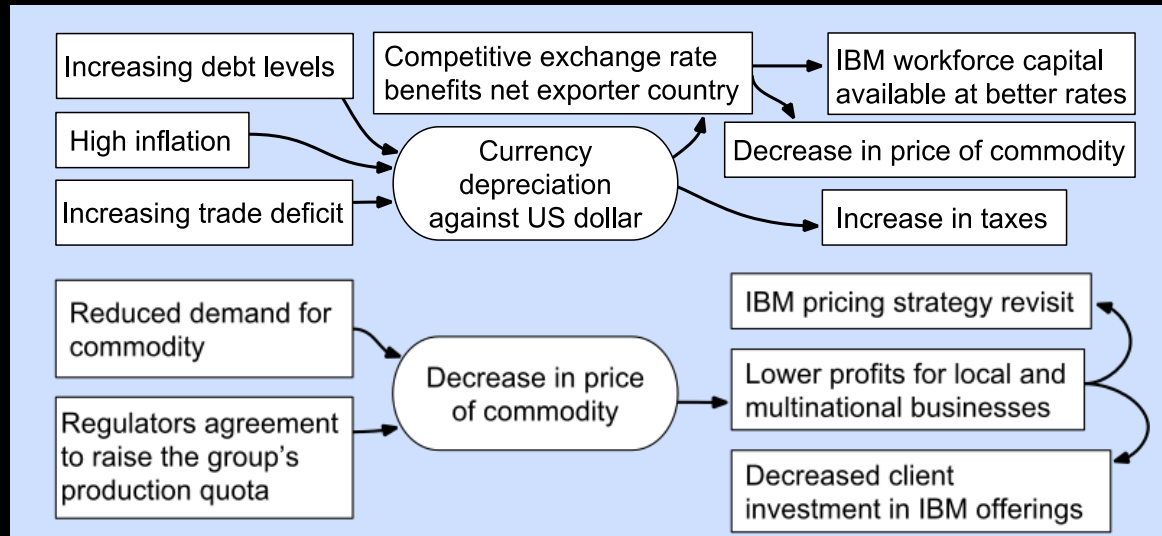
# Scenario Planning for Enterprise Risk Management

Main forces:

- Economic environment
- Technology
- Currency
- Social order/unrest
- Corruption
- Natural disaster
- Market disruptors
- Government Stability



# Mindmaps (in the context of Scenario Planning Advisor)



Question 1 of 4

How likely are any of the following to lead to **currency depreciation against US dollar**?

**High inflation**

**Likelihood**

High

**Increasing trade deficit**

Medium

# Planning Models for Scenario Planning – How Acquired?

**Manual creation of scenario planning models is impracticable.**

**Automated extraction of causal models** (Hassanzadeh et al. 2019, Bhandari et al., 2021)

- use learning-based natural language understanding techniques to identify risk drivers and extract causal pairs
- AI QA for automated reading comprehension
- with causal questions (“what causes X”) no supervision is needed
- authoritative documents that are rich in causal content (10-K, NATO SFA)
- with open-ended questions based on seed sets of risks, discover new candidate risks, bootstrap

# Causal Extraction in Support of Planning Model Generation

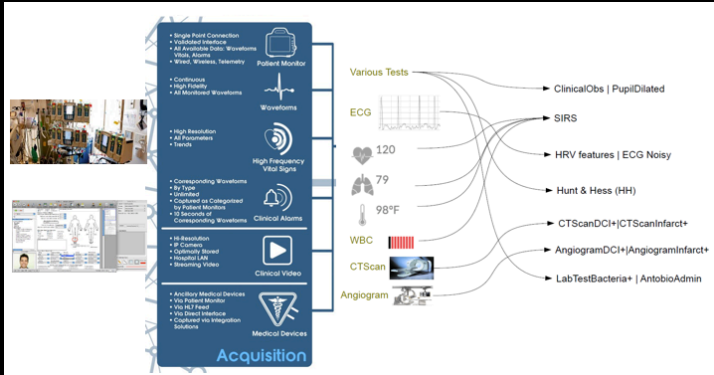
## **CEFAS – Causal Extraction From Authoritative Sources**

- uses AI Question Answering to answer causal questions about risk drivers...
- relies on relatively small set of authoritative documents, e.g.,
  - SEC Form 10-K – Item 1A Risk Factors
  - NATO Strategic Foresight Analysis doc, 2017
- CEFAS processes all input documents, asking of each paragraph of each document and for each known risk driver:

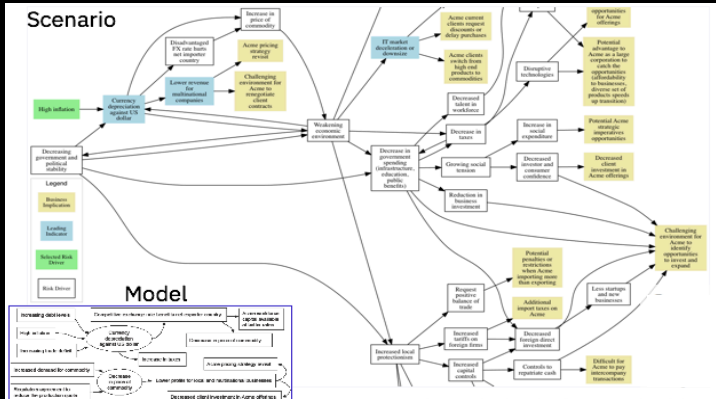
**What causes D?**

**What does D cause?**

# Examples



[Automated large-scale data analysis, ICAPS 2015]



25 [Scenario planning for enterprise risk management, AAI 2018]

[D3WA+: A Case Study of XAIP in a Model Acquisition Task, ICAPS 2020]



[Exploring Context-Free Languages via Planning: The Case for Automating Machine Learning, ICAPS 2020]

# State of Conversational Agents Today

- **Players:** Google's DialogueFlow, IBM's Watson Assistant, Apple's Siri, Amazon's Alexa, Microsoft's Cortana, etc.

**Agent:** Hello, how can I help you?

**User:** I'd like a trip to Toronto please.

**Agent:** Ok, a trip. Where to?

**User:** Ugh, never mind.

**Agent:** Hi. What seems to be the issue?

**User:** All of my emails are gone! Help!!

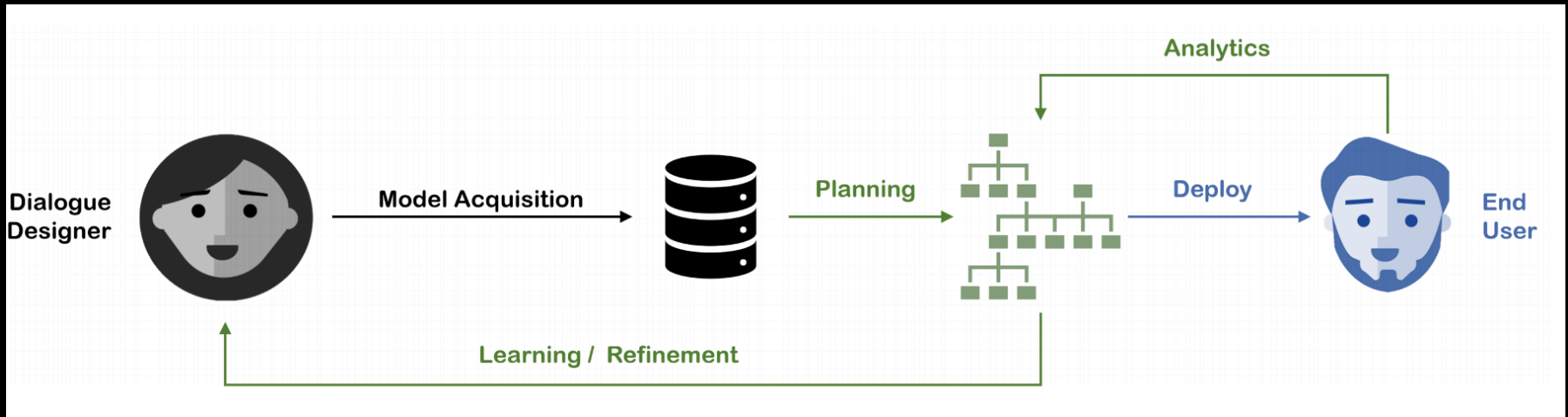
**Agent:** I understand entirely how frustrating this can be. We know exactly what happened, and will have it resolved in an hour. If you'd like to check the status, your ticket is #123.

**User:** That's a relief, thanks!

# Dialog as Planning

**Setting :** Multi-turn goal-oriented dialogue

**Challenge:** How to scale beyond explicit trees



Muise et al., 2019. Planning for Goal-Oriented Dialogue Systems.

Sreedharan et al., 2020. Explainable Composition of Aggregated Assistants

Rizk et al., 2020. A Unified Conversational Assistant Framework for Business Process Automation.

# D3WA Exponential scale-up of sophistication of composed conversations

Exponential scale-up means

Increased sophistication of dialogue tree with same size of specification

Decreased size of specification for same sophistication of dialogue tree

**Build your Bot** Save Load Import from WIA

Suggest 0%

Add new action Select start action start\_conversation Filter actions by Select Denote Filters

start\_conversi... ask-for-oil-level ask-for-clutch-s... ask-for-break-p...  
ask-for-spark-pl... end\_conversation state-message

**Abstraction for skills**

7 actions automatically compiled into the graph

Compose

speech actions  
API calls  
logical inferences

Detailed description: This screenshot shows the 'Build your Bot' interface. At the top, there are buttons for 'Save', 'Load', and 'Import from WIA'. Below is a search bar with 'Suggest 0%'. A section titled 'Add new action' shows a list of actions: 'start\_conversation', 'ask-for-oil-level', 'ask-for-clutch-s...', 'ask-for-break-p...', 'ask-for-spark-pl...', 'end\_conversation', and 'state-message'. A red arrow points from this list to a detailed view of the 'ask-for-break-p...' action, which is an abstraction for skills. Below this, a large, complex graph represents the dialogue tree, with a black arrow pointing from the text '7 actions automatically compiled into the graph' to it. A red arrow points from the graph to a circular icon labeled 'Compose', which contains icons for 'speech actions', 'API calls', and 'logical inferences'.

**Beta features** Build • Visualize • Chat • Deploy Import domain variables from existing dialogue agent in Watson Assistant

D3WA Home Model Import from WIA Meta Writer assist Beta feature Save Load Import from WIA

Bot Name: Car-Inspection

List of Variables: spark\_plug\_status, break\_pad\_status, user\_initiative\_message, clutch\_seal\_tightness\_status, pass\_status\_options, all\_status, user\_initiative, message

Bot app: Please check clutch seal tightness.

ADD new action Select start action start\_conversation Filter actions by Select Denote Filters

state-message start\_conversation ask-for-oil-level  
ask-for-break-pad ask-for-spark-pl... end\_conversation

Actions (collapsed) expanded

NEEDS or preconditions

Variables (collapsed)

UPDATEs to variables inside an outcome

Modeling Enhancements

In-house chat

Non-deterministic OUTCOMES

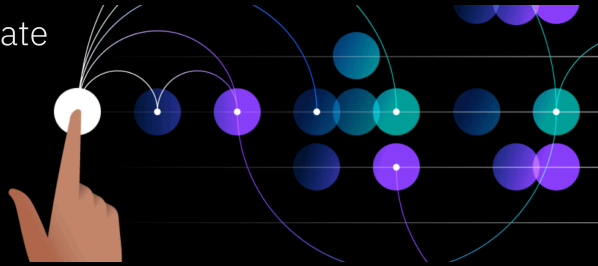
Detailed description: This screenshot shows a detailed view of the dialogue flow in the D3WA interface. At the top, there are buttons for 'Build', 'Visualize', 'Chat', and 'Deploy', and a link to 'Import domain variables from existing dialogue agent in Watson Assistant'. The interface shows a list of variables on the left, including 'spark\_plug\_status', 'break\_pad\_status', 'user\_initiative\_message', 'clutch\_seal\_tightness\_status', 'pass\_status\_options', 'all\_status', 'user\_initiative', and 'message'. The main area shows a dialogue flow starting with 'Bot app: Please check clutch seal tightness.' Below this, there are sections for 'ADD new action' and 'Define this dialog as a constant entity extraction?'. The dialogue flow includes actions like 'state-message', 'start\_conversation', 'ask-for-oil-level', 'ask-for-break-pad', 'ask-for-spark-pl...', and 'end\_conversation'. There are annotations for 'Actions (collapsed) expanded', 'NEEDS or preconditions', 'Variables (collapsed)', 'UPDATEs to variables inside an outcome', and 'Modeling Enhancements'. On the right, there is an 'In-house chat' window with buttons like 'Ready to record', 'Oil level is good', 'What's next', 'Please check break pads', 'What are the options', 'The options are pass and fail', 'Please check break pads', 'They pass', and 'OK, break pads pass'. At the bottom, it says 'Non-deterministic OUTCOMES'.



# Natural Language Processing x AI Planning

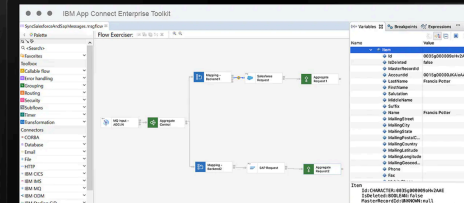
## IBM Watson Orchestrate

It's time to work differently by giving teams their own digital employee



## IBM App Connect

Connect your apps to automate your business



- Conversational agents realized as an aggregation of skill
- At runtime, planners can be used to organize assets dynamically to fulfill user goals extracted from natural language utterances.

- ICAPS 2021 Demo: <https://youtu.be/K7FPcl-IYgE>
- AAI 2022 Demo: [ibm.biz/gofa-aaai-demo](https://ibm.biz/gofa-aaai-demo)
- BPM 2021 Tutorial: [ibm.biz/bpm-2021-tutorial](https://ibm.biz/bpm-2021-tutorial)

# Related Work

- ItSimple (Vaquero et al., 2007)
- PDDL Editor <http://editor.planning.domains/>
- PDDL in Python <https://github.com/IBM/pddl-in-python>
- Planimation <https://github.com/AI-Planning/planimation>
- PDDL plugin for VSCode
- Workshop on Knowledge Engineering for Planning and Scheduling (KEPS) @ ICAPS
- PDDL book titled “*An Introduction to the Planning Domain Definition Language*” (Haslum et al., 2020)
- Extensive literature on planning and learning (Aineto et al., AIJ 2019)

# Modeling Summary

- Modeling is an important aspect of use and adaptation of AI Planning
- Investing in transformation/compilation techniques pays off
- Learning consumable planning models or even enhancing a planning model automatically is important