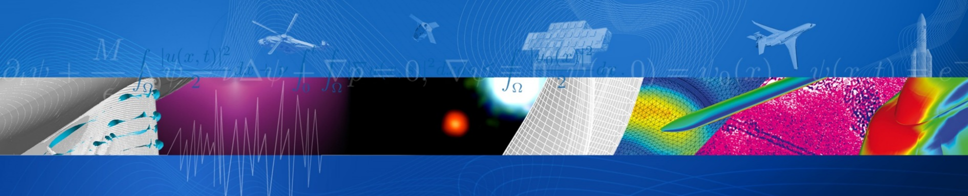


Contingent Planning using Counter-Examples from a Conformant Planner

Sébastien PIEDADE[1], Alban GRASTIEN[2],
Charles LESIRE[1], Guillaume INFANTES[3]
sebastien.piedade@onera.fr

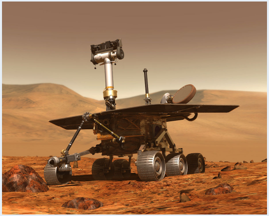
ONERA[1], DTIS, University of Toulouse, France
Australian National University[2], Canberra, Australia
JOLIBRAIN[3], Toulouse, France

ONERA
THE FRENCH AEROSPACE LAB

Context

- Search and rescue missions with local observations
- Uncertain environment
- Partial observability

- Plan → action sequence leading to a goal from an initial state
- Planning under uncertainty
- Find a plan allowing to reach a goal by performing observations of the environment



State of the art

- Replanning
  - Compute a first plan without considering uncertainties and replan if an unexpected event occurs during the execution of the plan
    - FF-Replan (Yoon et al., 2007)
- Uncertainty modeled by probabilities on state transitions and actions effects
  - Optimal policy applying the best action to each state
  - **Markov Decision Processes** (MDPs) (Puterman, 2014)
  - **Partially observable MDP** (Kaelbling et al., 1998)
    - Belief states and observations to update the belief
  - In the problems we are trying to solve, it is difficult to specify probabilities
- Uncertainty modeled by a set of possible initial states
  - **Conformant Planning** (Bertoli et al., 2001)
    - Compute a plan (an action sequence) solving the problem **whatever the possible initial states** without observation
    - Conformant-FF(Brafman and Hoffmann, 2004) , CPCES (Grastien et al., 2017)
  - **Contingent Planning** (Hoffmann and Brafman, 2005)
    - Compute a **Conditional plan** containing branches allowing an **online decision making** conditionned by the result of **observations**
    - Contingent-FF (Hoffmann and Brafman, 2005) , CLG (Albore et al., 2011)
    - Compilation approaches (Brafman and Shani, 2012)

Motivations

- Contingent planning has computation time issue
- The objective is to reduce this complexity
    - Compilation based approaches (Brafman and Shani, 2012)
- Reduces the contingent plan computation complexity by using a conformant planner
- Allows subproblems computations without considering the costly computation of observations

Approach

Idea

- CPCES Conformant Planner (Grastien et al., 2017), in case of failure:
    - Returns a **counter-example**
    - Returns a **plan solution of some of the possible initial states**
- Give the problem to solve to CPCES
- In case of failure, use the **counter-example** and the **failing plan** to determine **which observation** perform and **when**
- Split the problem in **subproblems** taking into account the observation to **reduce uncertainty** in the subproblems
- Use the Conformant Planner to **iteratively** solve these subproblems

Plan

1. Formalism
2. Algorithm Principle
3. Results
4. Conclusion
5. Approach Improvements

Formalism

Problem $(\mathcal{L}, \mathcal{O}, I, G)$

- $\mathcal{L} = \{p_1, ..., p_n\}$ is a finite set of **propositions**
- $\mathcal{W} = 2^{\mathcal{L}}$ is the set of possible **world states**
- $\mathcal{O}$ is a finite set of **operators**, divided in **action** sets A and **observation** sets O
- Each operator $op \in \mathcal{O}$ is defined by **preconditions** $pre(op) \subseteq \mathcal{L}$ and a **set of effects** $eff(op)$
- $I \subseteq \mathcal{W}$ is the set of **possible initial states**
- $G \subseteq \mathcal{L}$ is the set of propositions defining the **goal**

Formalism

Action application in a world state

- $a \in A$ is *applicable* in the state $s \in \mathcal{W}$ iff $pre(a) \subseteq s$
- An effect $e \in eff(a)$ is defined by a triplet $(con(e), add(e), del(e))$
- If a is applicable in s, then the application of a results in a state $T(s, a)$ with T the **transition function** such that:

$$T(s,a) = s - \bigcup_{e \in eff(a) \text{ s.t. } con(e) \subseteq s} del(e) \cup \bigcup_{e \in eff(a) \text{ s.t. } con(e) \subseteq s} add(e)$$

Application of an observation in a world state

- $o \in O$ is applicable in $s \in \mathcal{W}$ iff $pre(o) \subseteq s$
- $eff(o) \in \mathcal{L}$ is defined by the **truth value** of the observed proposition
- The application of o in s has **no effect** on s, T(s,o) = s

Formalism

Belief state

- **Belief** $\mathcal{B} = \{s_1, \ldots, s_n\} \subseteq \mathcal{W}$ is the propagation of the possible initial states

Action and Observation application in a Belief

- An action $a \in A$ is applicable in a belief $\mathcal{B}$ if a is applicable in each state of $\mathcal{B}$. The application of a in $\mathcal{B}$ consists in applying a to each state of $\mathcal{B}$
- An observation $o \in O$ is applicable in a belief $\mathcal{B}$ if o is applicable in each state of $\mathcal{B}$. The application of o in $\mathcal{B}$ results in a new belief in which the states that does not contain the observed proposition are removed

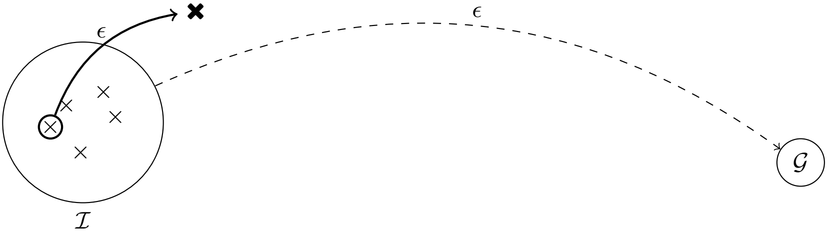
Conditionnal Plan

- **Operators Graph**, leading an initial belief to a **goal** belief
- **Branchings** in the graph corresponds to **observations results**, whether the observed properties are in the current belief or not

Plan

1. Formalism
2. Algorithm Principle
3. Results
4. Conclusion
5. Approach Improvements

CPCES (Grastien et al., 2017)



CPCES checks if the empty plan is solution

Legend

×	possible initial state	- - - - →	plan proposed to Z3
○⟶✖	counter-example found by Z3	⟶	plan computed by FF
$\boxed{a}$	returned element		

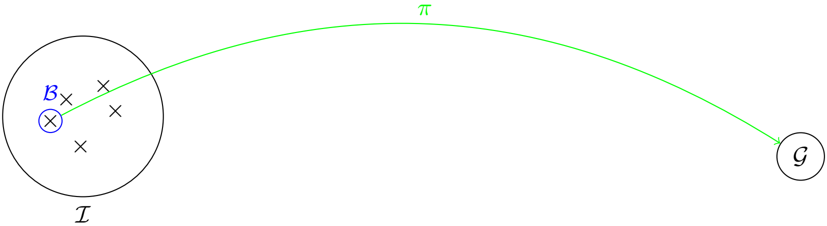
CPCES (Grastien et al., 2017)

CPCES selects a counter-example state in which the empty plan fails

Legend

×	possible initial state	- - - - →	plan proposed to Z3
○⟶ ✖	counter-example found by Z3	⟶	plan computed by FF
$\boxed{a}$	returned element		

CPCES (Grastien et al., 2017)

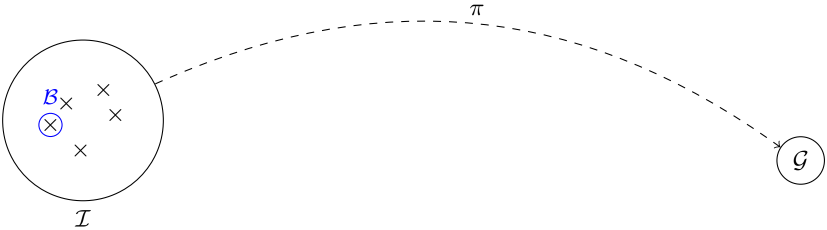


FF is used to compute a plan π from the set of considered states $\mathcal{B}$

Legend

×	possible initial state	- - - - →	plan proposed to Z3
○⟶✖	counter-example found by Z3	⟶	plan computed by FF
$\boxed{a}$	returned element		

CPCES (Grastien et al., 2017)

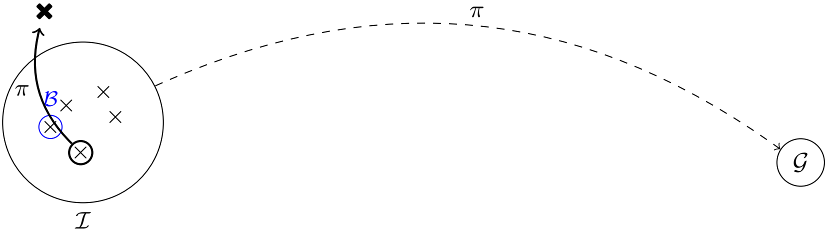


The SAT-solver Z3 checks if π is valid for the complete set of initial states

Legend

×	possible initial state	- - - - →	plan proposed to Z3
○⟶✖	counter-example found by Z3	⟶	plan computed by FF
$\boxed{a}$	returned element		

CPCES (Grastien et al., 2017)



If π is valid then it is returned, else a counter-example is found

Legend

×	possible initial state	- - - - →	plan proposed to Z3
○ ⟶ ✖	counter-example found by Z3	⟶	plan computed by FF
$\boxed{a}$	returned element		

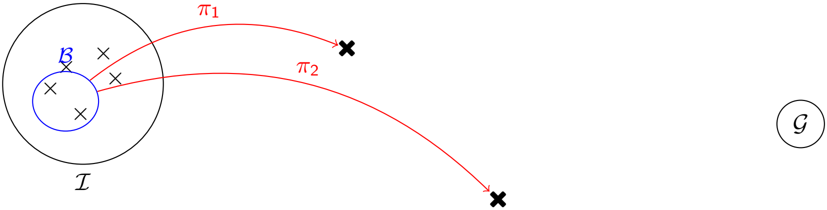
CPCES (Grastien et al., 2017)

CPCES adds the counter-example to $\mathcal{B}$ and iterates until a conformant plan is found

Legend

×	possible initial state	- - - - →	plan proposed to Z3
○⟶ ✖	counter-example found by Z3	⟶	plan computed by FF
$\boxed{a}$	returned element		

CPCES (Grastien et al., 2017)

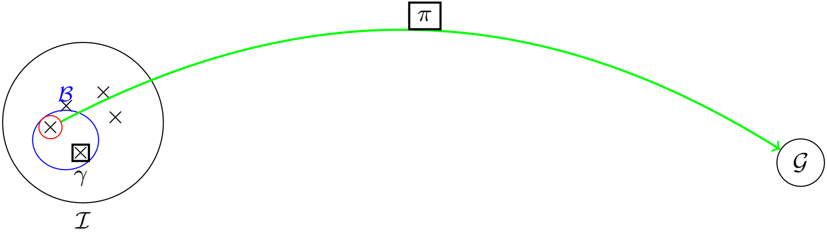


If FF cannot compute a plan, then there is no conformant plan

Legend

×	possible initial state	- - - - →	plan proposed to Z3
○⟶ ✖	counter-example found by Z3	⟶	plan computed by FF
$\boxed{a}$	returned element		

CPCES (Grastien et al., 2017)

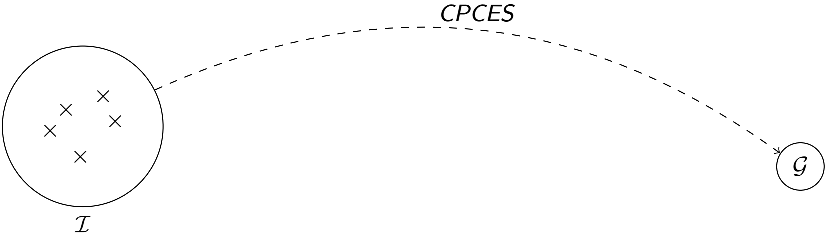


The last counter-example and the previous plan are then returned

Legend

×	possible initial state	- - - - →	plan proposed to Z3
○⟶✖	counter-example found by Z3	⟶	plan computed by FF
$\boxed{a}$	returned element		

Contingent Process

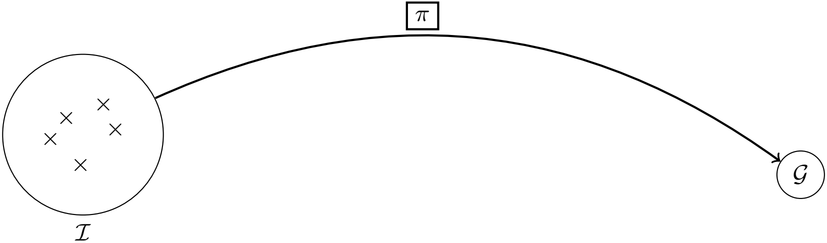


CPCES is called to compute a conformant plan

Legend

×	state	→	success
- - - →	application	$\boxed{a}$	returned element
→	failure		

Contingent Process

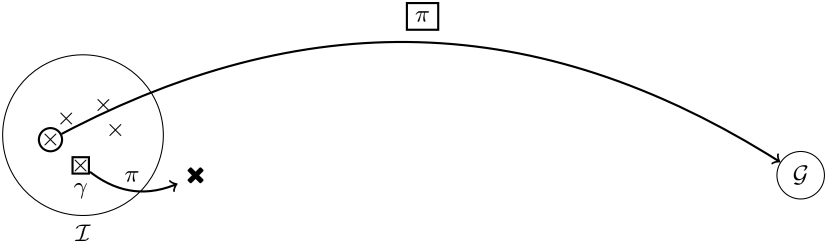


If CPCES finds a conformant plan then it is returned

Legend

×	state	→	success
- - - - →	application	$\boxed{a}$	returned element
→	failure		

Contingent Process

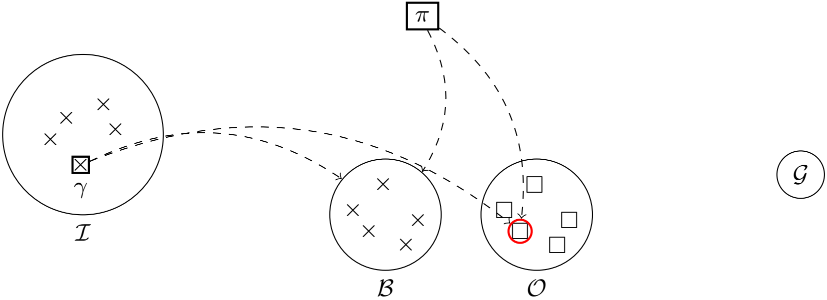


Else a counter-example and a failing plan are extracted

Legend

×	state	⟶	success
- - - - →	application	$\boxed{a}$	returned element
⟶	failure		

Contingent Process

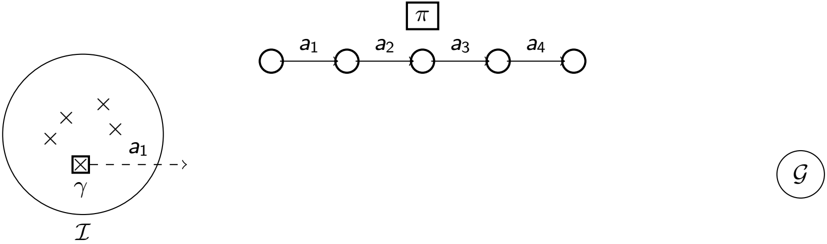


Identify the observation to perform and in which belief

Legend

×	state	→	success
- - - - →	application	$\boxed{a}$	returned element
→	failure		

Contingent Process

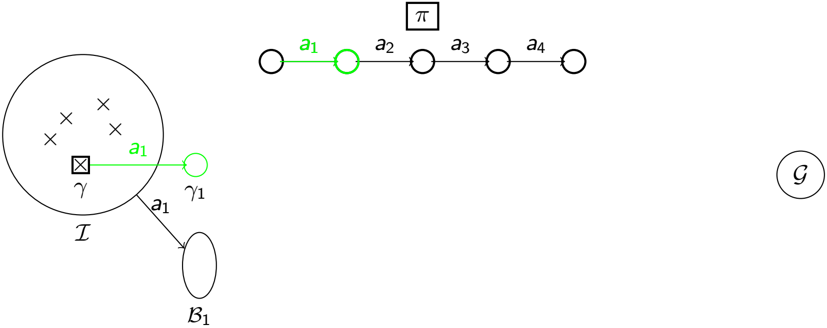


Identification of the action of π that fails in γ

Legend

×	state	$\longrightarrow$	success
$- - - - \rightarrow$	application	$\boxed{a}$	returned element
$\longrightarrow$	failure		

Contingent Process

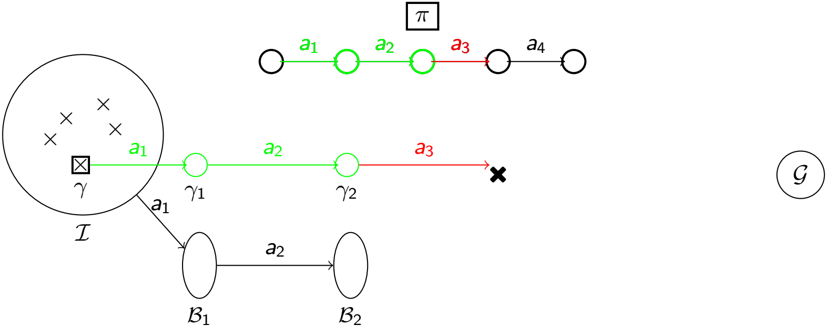


If the action is applicable, apply it to $\mathcal{I}$ to keep track of the beliefs

Legend

×	state	⟶	success
- - - - →	application	$\boxed{a}$	returned element
⟶	failure		

Contingent Process

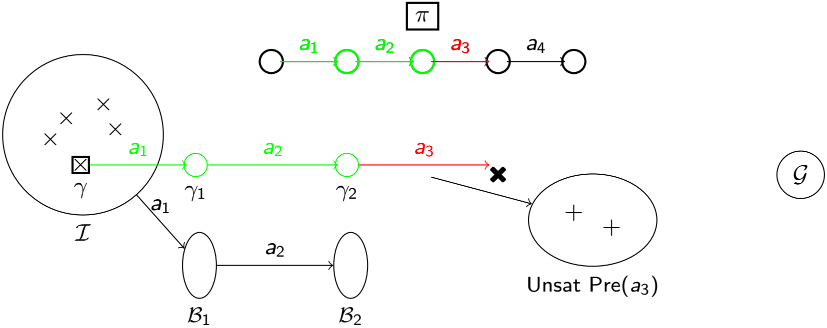


The process iterates until an action is not applicable

Legend

×	state	→	success
- - - - →	application	$\boxed{a}$	returned element
→	failure		

Contingent Process

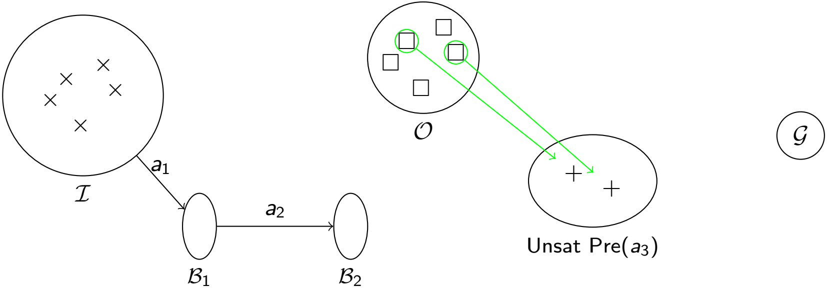


Extract the propositions in the preconditions of a_3 that fail

Legend

×	state	⟶	success
- - - - →	application	$\boxed{a}$	returned element
⟶	failure		

Contingent Process

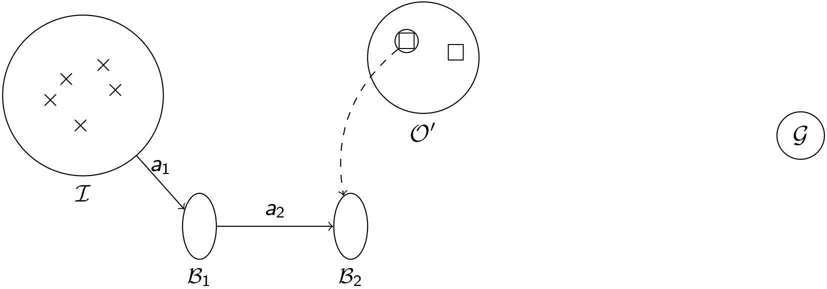


Extract the observations able to observe one of the unsatisfied propositions

Legend

×	state	→	success
- - - - →	application	$\boxed{a}$	returned element
→	failure		

Contingent Process

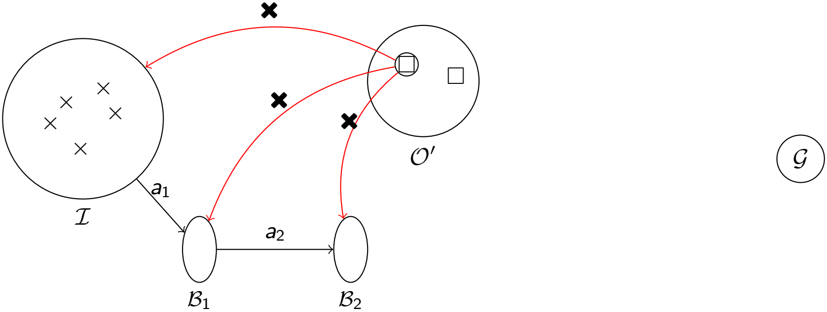


Apply a first observation to each belief

Legend

×	state	→	success
- - - →	application	$\boxed{a}$	returned element
→	failure		

Contingent Process

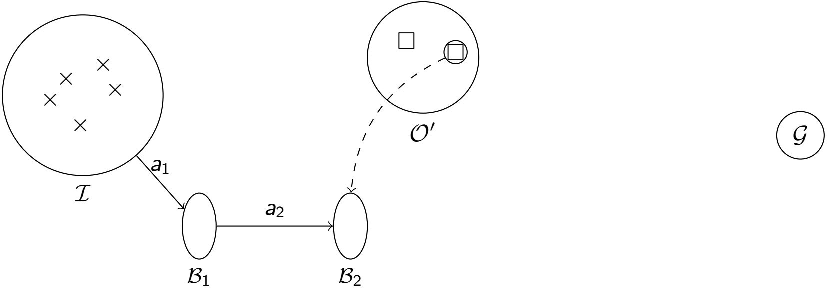


The current observation is not applicable in any belief

Legend

×	state	→	success
⇢	application	$\boxed{a}$	returned element
→	failure		

Contingent Process

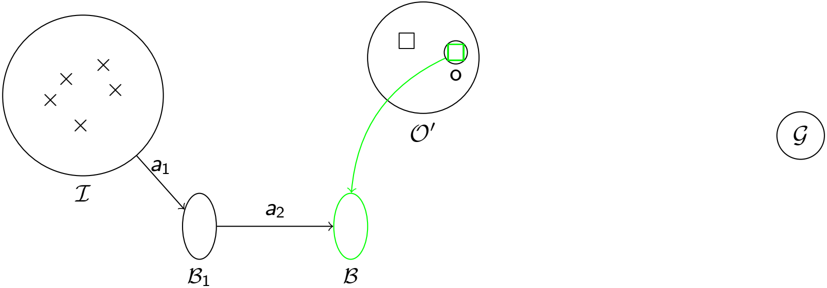


Try another observation

Legend

×	state	→	success
⇢	application	$\boxed{a}$	returned element
→	failure		

Contingent Process

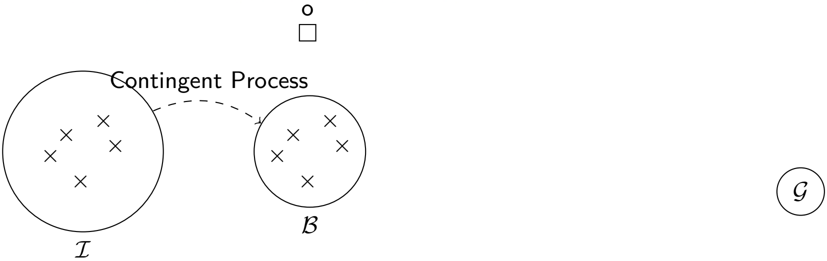


If the observation is applicable, the observation and belief are extracted

Legend

×	state	⟶	success
- - - - →	application	$\boxed{a}$	returned element
⟶	failure		

Contingent Process

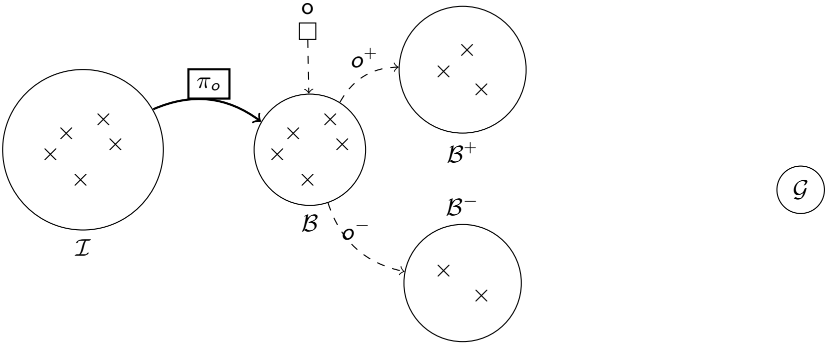


Restart the Contingent process to compute a plan leading to the observation

Legend

×	state	⟶	success
- - - →	application	$\boxed{a}$	returned element
⟶	failure		

Contingent Process

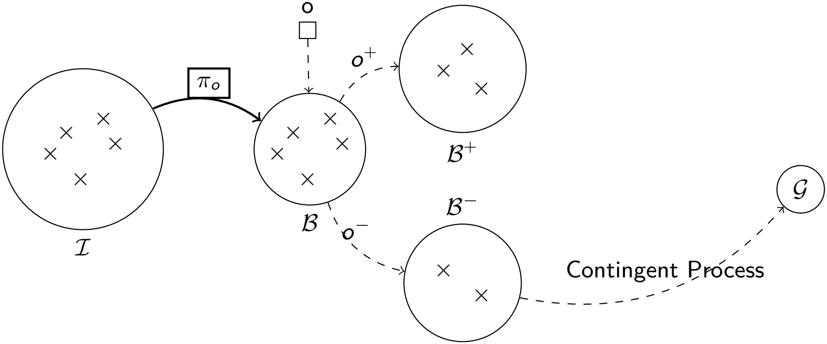


Split the belief considering the observation

Legend

×	state	⟶	success
- - - - →	application	$\boxed{a}$	returned element
⟶	failure		

Contingent Process

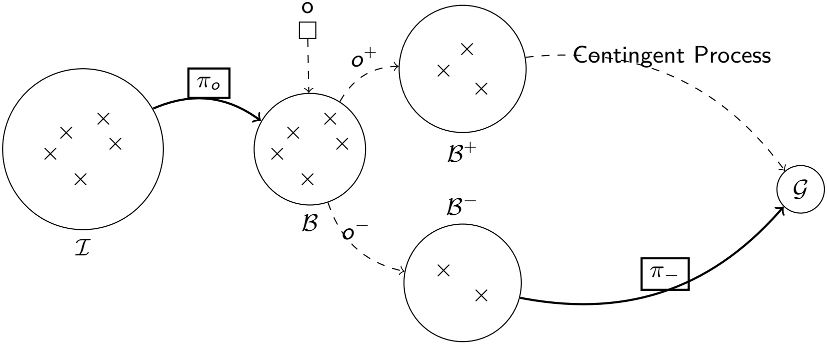


Start the Contingent process to compute the branch from $\mathcal{B}^-$ to the goal

Legend

×	state	⟶	success
- - - - →	application	$\boxed{a}$	returned element
⟶	failure		

Contingent Process

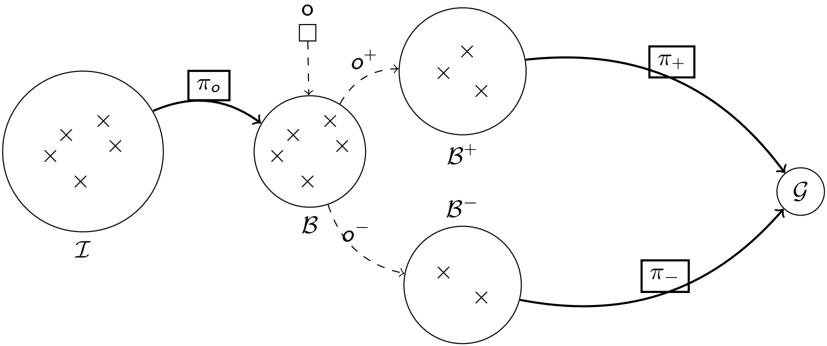


Start the Contingent process to compute the branch from $\mathcal{B}^+$ to the goal

Legend

×	state	🟩⟶	success
- - - →	application	$\boxed{a}$	returned element
🟥⟶	failure		

Contingent Process

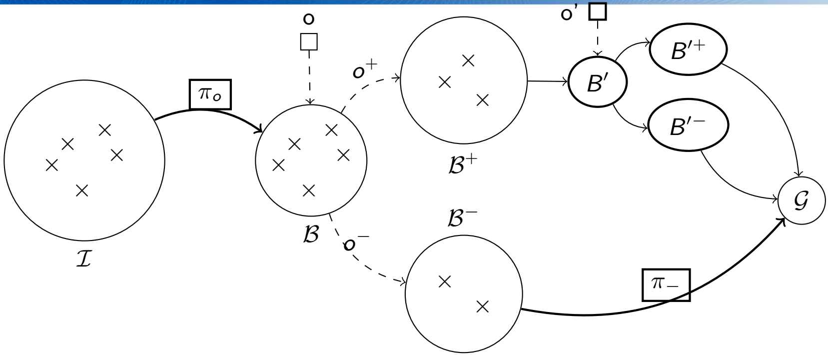


Once each branch leads to the goal, the resulting Contingent plan is returned

Legend

×	state	→	success
- - - →	application	$\boxed{a}$	returned element
→	failure		

Contingent Process



The process is recursive: a subplan can be Conformant, or Contingent

Legend

×	state	⟶	success
- - - - →	application	$\boxed{a}$	returned element
⟶	failure		

Contingent Planning Algorithm

Algorithm 1: Contingent Planning Procedure

```
Input: $\mathbb{P} = (\mathcal{L}, \mathcal{O}, I, G)$
Output: $\pi_c$
begin
    $\pi, \gamma := \mathrm{conformantPlanner}(\mathbb{P})$
    if $\gamma = \emptyset$ then
        return $\pi$
    $\mathcal{B}_o, o := \mathrm{findObservation}(I, \mathcal{O}, \pi, \gamma)$
    $\pi_o := \mathrm{ContingentPlanning}((\mathcal{L}, \mathcal{O}, I, \mathcal{B}_o))$
    $\mathcal{B}^+ := T(\mathcal{B}_o, o)$ with $\nu^+(o) = eff(o)$
    $\pi_p := \mathrm{ContingentPlanning}((\mathcal{L}, \mathcal{O}, \mathcal{B}^+, G))$
    $\mathcal{B}^- := T(\mathcal{B}_o, o)$ with $\nu^-(o) = eff(o)$
    $\pi_n := \mathrm{ContingentPlanning}((\mathcal{L}, \mathcal{O}, \mathcal{B}^-, G))$
    return $(\pi_o$ ; if $o$ then $\pi_p$ else $\pi_n)$
```

- The algorithm is described in the paper
- Our approach is sound because we rely on a sound conformant planner to compute each branch of the plan
- The returned solutions are conformant or contingent solutions
- We compare our approach with Contingent-FF (Hoffmann and Brafman, 2005)
- 7 benchmarks of 4 problems each
- Time out of 5 min

Plan

1. Formalism
2. Algorithm Principle
3. **Results**
4. Conclusion
5. Approach Improvements

Results

Problem	Contingent Planning with counter-examples					Contingent-FF			
	time	size	depth	shortest	observations	time	size	depth	observations
blocks/p3	0.94	6	4	3	1	0.00	6	4	1
blocks/p7	5.6	89	16	10	7	0.05	**55**	**9**	7
blocks/p11	6.4	169	29	20	7	0.43	**117**	**18**	7
blocks/p15	8.05	244	39	27	7	3.20	**163**	**25**	7
btcs/p10	0.76	19	19	19	**0**	0.02	19	10	9
btcs/p30	2.36	59	59	59	**0**	0.8	59	30	29
btcs/p50	8.13	99	99	99	**0**	9.79	99	50	49
btcs/p70	24.11	139	139	139	**0**	57.31	139	70	69
ebtcs/p10	6.21	19	10	2	9	0.01	19	10	9
ebtcs/p30	22.73	59	30	2	29	0.42	59	30	29
ebtcs/p50	56.8	99	50	2	49	4.93	99	50	49
ebtcs/p70	156.11	139	70	2	69	29.10	139	70	69
grid/p2	3.61	9	9	9	0	0.01	9	9	0
grid/p3	4.05	**19**	**19**	19	**0**	9.78	174	43	15
grid/p4	21.24	**45**	**45**	45	**0**	227	464	68	17
grid/p5	18.64	**31**	**31**	**31**	**0**	TO	-	-	-
erovers/p2	1.09	**11**	**9**	5	1	0.00	13	10	1
erovers/p4	3.39	23	17	5	3	0.00	23	**14**	3
erovers/p6	NO	-	-	-	-	0.09	**346**	**48**	**7**
erovers/p8	3.34	**44**	**21**	15	3	0.01	95	36	3
logistics/p1	0.39	**9**	9	9	**0**	0.01	10	7	1
logistics/p3	0.49	**14**	14	14	**0**	0.01	18	8	2
logistics/p5	0.58	**29**	29	29	**0**	0.054	172	26	7
logistics/p7	0.75	**31**	31	31	**0**	0.2	247	27	11
elogistics/p1	1.07	10	7	4	1	0.00	10	7	1
elogistics/p3	1.8	18	8	5	2	0.00	18	8	2
elogistics/p5	9.02	**138**	**22**	20	7	0.12	172	26	7
elogistics/p7	10.63	**185**	26	21	11	0.13	247	26	11

Results

- Generally we find **shorter plans**
- When a **conformant solution** exists:
  - We find a **conformant plan** while Contingent-FF (Hoffmann and Brafman, 2005) include **observations**
- When there is no conformant solution:
  - Generally the **same number of observations** but **shorter plans**
- Drawbacks:
  - Lack of completeness
  - Computation time

Plan

1. Formalism
2. Algorithm Principle
3. Results
4. **Conclusion**
5. Approach Improvements

Conclusion

- Contingent Planner computing contingent plans with a limited complexity
- Use a conformant planner to compute **conformant subplans** if possible
- Use a **counter-example** and a **failing plan** to find an **observation**
- Split the problem in **subproblems** with **less uncertainty**
- Iterate until no counter-example exists
- Any conformant planner can be used if it returns a counter-example

Plan

1. Formalism
2. Algorithm Principle
3. Results
4. Conclusion
5. Approach Improvements

Approach improvements

Huge Space Version

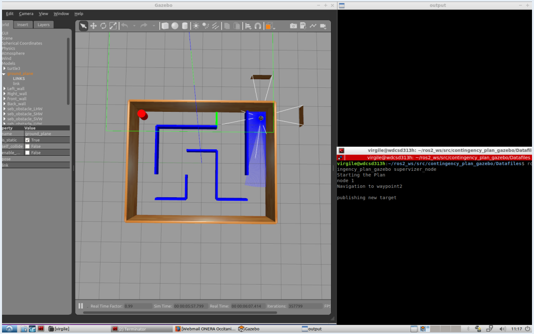
- We do not compute the beliefs internally, we directly modify the PDDL problem and domain
- We ask CPCES (Grastien et al., 2017) to find a plan leading to one of the possible observations
- To compute the branches we enforce CPCES to start with the plan leading to the observation and we transmit a SMT contraint to indicate the result of each observation contained in the previous branch
- Ongoing results

Backtracking

- Backtracking on the observations
- Handle a list of the plans computed from every possible observations
- Improve the completeness of the approach
- Ongoing results

Simulation Environment

- Simulation environment of robotic mission
- Internship of Virgile De La Rochefoucauld at the Onera



```
virgile@wdcsd313h: ~/ros2_ws/src/contingency_plan_gazebo/Datafil
virgile@wdcsd313h: ~/ros2_ws/src/contingency_plan_gazebo/Datafiles
virgile@wdcsd313h:~/ros2_ws/src/contingency_plan_gazebo/Datafiles$ r
ingency_plan_gazebo supervizer_node
Starting the Plan
node 1
Navigation to waypoint2

publishing new target
```

Albore, A. et al. (2011).
Translation-based approaches to automated planning with incomplete information and sensing.
PhD thesis, Universitat Pompeu Fabra.

Bertoli, P., Cimatti, A., and Roveri, M. (2001).
Heuristic search+ symbolic model checking= efficient conformant planning.
In IJCAI, pages 467–472. Citeseer.

Brafman, R. and Hoffmann, J. (2004).
Conformant planning via heuristic forward search: A new approach.
In International Conference on Automated Planning and Scheduling (ICAPS), Whistler, Canada.

Brafman, R. and Shani, G. (2012).
A multi-path compilation approach to contingent planning.
In Twenty-Sixth AAAI Conference on Artificial Intelligence.

Grastien, A., Scala, E., and Kessler, F. B. (2017).
Intelligent belief state sampling for conformant planning.
In Proceedings of the 26th International Joint Conference on Artificial Intelligence, pages 4317–4323. AAAI Press.

Hoffmann, J. and Brafman, R. (2005).
Contingent planning via heuristic forward search with implicit belief states.
In Proc. ICAPS, volume 2005.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1998).
Planning and acting in partially observable stochastic domains.
Artificial intelligence, 101(1-2):99–134.

Puterman, M. L. (2014).
Markov decision processes: discrete stochastic dynamic programming.
John Wiley & Sons.

Yoon, S. W., Fern, A., and Givan, R. (2007).
Ff-replan: A baseline for probabilistic planning.
In ICAPS, volume 7, pages 352–359.