Fault Tolerant Planning: Complexity and Compilation

Carmel Domshlak
Technion
Haifa, Israel
dcarmel@ie.technion.ac.il

Abstract
In the context of modeling and reasoning about agent actions, contingent and classical planning can often be respectively seen as adopting “extreme pessimism” and “extreme optimism” about the action outcomes. For many everyday scenarios of human reasoning (and thus for many types of autonomous systems), both these approaches are just too extreme. Following Jensen, Veloso, and Bryant (2004), we examine a planning model that interpolates between classical and contingent planning via tolerance to arbitrary faults occurring during plan execution. We show that an important fragment of this fault tolerant planning (FT-planning) exhibits both an appealing solution structure, as well as appealing worst-case time-complexity properties. We also show that such FT-planning tasks can be efficiently compiled into classical planning as long as the number of possible faults per operator is bounded by a constant, and we show that this compilation can be attractive in practice.

Introduction
To date, contingent and classical planning appear to be the two major approaches to non-probabilistic planning under full observability. In contingent planning, at least some aspects of system dynamics are modeled by operators with non-deterministic effects, and a plan should guarantee reaching a goal state under any realization of the actions it prescribes. In classical planning, the operators are all set to be deterministic, modeling only the singular intended effects of each action. While contingent plans provide much stronger guarantees on reaching the goal with respect to the true physics of the modeled system, they are also much harder to generate (both worst-case and empirically), and quite often they may simply not exist.

In the physical world, no actions are really guaranteed to succeed. However, non-determinism in real-world domains is often caused by infrequent errors that make otherwise deterministic operators fail. Hence, many unsolvable contingent planning tasks become solvable if we assume that no more than some $\kappa$ exceptional/faulty action effects will occur along the purposed plan to the goal. In the past, this observation brought numerous researchers to consider explicit representation and reasoning about faults of agents’ actions (Georgeff and Lansky 1986; Williams et al. 2003; Giunchiglia, Spalazzi, and Traverso 1994). In particular, Jensen, Veloso, and Bryant (2004) suggested a model of fault tolerant planning (FT-planning), and developed first algorithms for generating plans that are robust for a single fault occurring during plan execution. This model is of our focus here.

Departing from contingent planning and generalizing the FT-planning model of Jensen, Veloso, and Bryant (2004), we show that, while FT-planning remains in general as computationally hard as contingent planning, one of its practically most valuable fragments, namely the one considered by Jensen, Veloso, and Bryant (2004), (1) is in PSPACE, (2) falls into NP when restricted to plans with only polynomial-length executions, and (3) is guaranteed to admit stationary solutions for solvable problems, solutions that sometimes induce (possibly cyclic) strong contingent plans. Furthermore, we show that these FT-planning tasks can be efficiently compiled into equivalent classical planning tasks in a way that is sound, complete, and practicable.

Our results join a growing body of work on planning under uncertainty and/or partial observability via compilation to classical planning (Palacios and Geffner 2009; Albore, Palacios, and Geffner 2009; Bonet and Geffner 2011; Brafman and Shani 2012a; Taig and Brafman 2013). At a high level, FT-planning is an instance of “assumption-based planning,” and the latter term has already been used for a broad range of ideas and techniques (Albore and Bertoli 2004; 2006; Albore and Geffner 2009; Bonet and Geffner 2011; Göbelbecker, Gretton, and Dearden 2011; Davis-Mendelow, Baier, and McIlraith 2012). Closest in spirit to our work here—though in two different ways—are probably the works of Albore and Bertoli (2006) and Davis-Mendelow, Baier, and McIlraith (2012). Albore and Bertoli suggested an interesting planning approach in which assumptions about operator effects are provided a priori as a linear temporal logic formula, and the planner takes these assumptions as axioms. In the worst-case, however, this approach remains as hard as contingent planning. Davis-Mendelow et al. exploit assumption-based assertions about the initial state to suggest a middle-ground between classical planning and conformant, or zero observability, planning. The latter, however, is very different from contingent planning, both conceptually and complexity-wise (Bonet 2010).
Planning Formalisms and Solution Concepts

Non-deterministic planning tasks with full observability correspond to succinctly represented, goal-oriented non-deterministic Markov decision processes (Puterman 1994). Several languages for succinctly representing such tasks are in use (Hoffmann and Brafman 2005; Bonet 2010; Davis-Mendelow, Baier, and McIlraith 2012). To simplify presentation, here we adopt a minimalist extension of STRIPS (Fikes and Nilsson 1971) to non-deterministic operator effects.

A **planning task** is given by a quadruple \( \Pi = \langle P, O, s_0, G \rangle \). \( P \) is a set of \( n \) propositions, with world states \( S \) being represented by complete valuations of \( P \), and usually discussed as sets of propositions that hold true in them. \( s_0 \in S \) is an initial state\(^1\), and \( G \) is a subset of \( P \): a state \( s \) is a goal state iff \( G \subseteq s \), and the set of all goal states is denoted by \( S_G \). \( O \) is a set of operators \( \Theta = \langle \text{pre}(\alpha), \text{eff}(\alpha) \rangle \) where the *precondition* \( \text{pre} \) is a subset of propositions \( P \), and \( \text{eff} = \{ e_1, \ldots, e_m \} \) is a set of possible effects of \( \alpha \). Each possible effect \( e \in \text{eff} \) is given by a pair \( (\text{add}(e), \text{del}(e)) \) of subsets of \( P \), corresponding to its add and delete lists, respectively. An operator \( \Theta \) is applicable in state \( s \) iff \( \text{pre}(\Theta) \subseteq s \), and the set of all such operators is denoted by \( O(s) \). If \( o \in O(s) \) is applied in \( s \), it changes the world to one of the states \( \text{Res}(s; o) = \bigcup_{e \in \text{eff}(o)} \{ \text{Res}(s; e) \} \), where \( \text{Res}(s; e) = (s \setminus \text{del}(e)) \cup \text{add}(e) \) is the state resulting from the effect \( e \) occurring in \( s \). A (contingent) **plan** for a task \( \Pi \) is an action strategy that guarantees reaching a goal state \( s \in S_G \) from \( s_0 \), under any realization of the operators applied along the way; the process of search for contingent plans is called **contingent planning**.

While nondeterministic operators must be dealt with in one form or another in many planning applications, two problems particular to contingent planning must be taken into account. First, deciding whether a contingent plan exists is EXPTIME-complete (Rintanen 2004).\(^2\) Second, many tasks admit no contingent plans, and this is true even for simple tasks that humans feel comfortable dealing with (Cimatti et al. 2003; Pistore and Vardi 2007). In that respect, a pragmatic alternative to contingent planning is **classical planning**, operators are deterministic. By adopting classical planning as an abstraction of contingent planning, we assume that we know precisely what will happen when an operator \( \Theta \) is applied in state \( s \). This assumption is then “encoded” at the level of individual operators by what is called determinization, reducing the set of possible effects of each operator to exactly one effect.

While being much more restricted, classical planning resolves to a large extent the two aforementioned shortcomings of contingent planning. First, restricting each state/action pair to a sole possible successor often renders unsolvable problems solvable. Second, classical planning is in \( \text{PSPACE} \) (Bylander 1994), and more importantly, it is in \( \text{NP} \) if restricted to polynomial-length plans. Last but not least, classical plans are structurally simple, constituting linear sequences of operators. Together with \( \text{NP} \)-membership, this structural simplicity allows for exploiting various OR-graph search techniques for developing empirically efficient solvers for classical planning (Hoffmann and Nebel 2001; Helpert 2006; Rintanen, Heljanko, and Niemelä 2006; Kissmann and Edelkamp 2012). As a result, combining classical planning with online re-planning in unexpected situations is a popular and effective approach to closed-loop control of autonomous systems (Yoon et al. 2008; Talamadupula et al. 2010; Domshlak et al. 2011; Bonet and Geffner 2011; Brafman and Shani 2012b).

**Fault Tolerant (Contingent) Planning**

Given the relative pros and cons of contingent and classical planning, the first question one might ask is: If these are the two extremes, how can we interpolate between them in a simple and useful manner? This question brought us to consider **fault tolerant planning**: planning under the assumption that no more than some \( \kappa \) unintended effects of the operators will occur along the purported plan to the goal, but at the same time, under a requirement for the plans to be provably robust for up to \( \kappa \) such operator faults during plan execution. Fault tolerant planning was originally introduced by Jensen, Veloso, and Bryant (2004) in order to bring some key information from probabilistic uncertainty models to qualitative non-deterministic planning. The basic idea is to associate the contingent planning task at hand with an explicit distinction between the primary and exceptional effects of its operators. The model we adopt for that purpose is simply a function \( F \) that maps each possible effect of each operator to the “number of exceptions,” or unintended artifacts, associated with this effect. The operator effects \( e \) for which \( F(e) = 0 \) correspond to the primary effects of the respective operators in \( s \).

**Definition 1** Let \( \Pi = \langle P, O, s_0, G \rangle \) be a contingent planning task. An **exception model** for \( \Pi \) is a function \( F : \bigcup_{\Theta \in O} \text{eff}(\Theta) \rightarrow \mathbb{N} \), computable in time polynomial in |\( \Pi \)|. If, for each operator \( \Theta \in O \), |\( \{ e \in \text{eff}(\Theta) \} | e \in F(\Theta) \}| \leq \alpha \), then \( F \) is called **\( \alpha \)-primary**. Likewise, if, for each operator \( \Theta \in O \), |\( \{ e \in \text{eff}(\Theta) \} | e \in F(\Theta) \}| > \alpha \), then \( F \) is called **\( \alpha \)-normative**.

In simple terms, an exception model is \( \alpha \)-primary if at most \( \alpha \) effects of each operator are considered to be its primary effects, and it is normative if each operator is associated with at least one primary effect. In these terms, the work of Jensen, Veloso, and Bryant (2004) has been devoted to fault tolerant planning under \( 1 \)-primary normative exception models, which seem to cover well operator non-determinism that stems from physical complications of executing agent actions in the real world.\(^3\) Associating non-deterministic operators with exception models allows for a

---

\(^1\)We assume here that there is no uncertainty about the initial state, and later discuss the impact of this assumption.

\(^2\)EXPTIME-completeness still holds even for testing the existence of plans that reach the goal with probability exceeding \( p \) for probabilistic problems with full observability (Littman 1997).

\(^3\)If, however, some operators model knowledge acquisition, i.e., sensing, then (part of the) operator non-determinism will be due to primary operator effects, and thus planning with \( \alpha \)-primary models for \( \alpha > 1 \) is not of theoretical interest only.
simple relaxation of contingent planning to planning under fault tolerance requirements as above. Let \( \Pi \) be a contingent planning task, \( \mathcal{F} \) be an exception model for \( \Pi \), and \( \pi \) be an action policy for \( \Pi \). Overloading the notation, for an execution \( \rho = (s_0, e_0, \ldots, s_i, e_i, \ldots) \) of \( \pi \), we define \( \mathcal{F}(\rho) = \sum_{i=0}^{\infty} \mathcal{F}(e_i) \).

- An execution \( \rho \) of \( \pi \) is called \( \kappa \)-admissible if \( \mathcal{F}(\rho) \leq \kappa \).
- Action policy \( \pi \) is a \( \kappa \)-plan for \( \Pi \) if each of its \( \kappa \)-admissible executions is finite and reaches the goal.

In what follows, we refer to triplets \( \langle \Pi, \mathcal{F}, \kappa \rangle \) as above as fault tolerant (FT) planning tasks, and solutions for such tasks are precisely \( \kappa \)-plans for \( \Pi \) under \( \mathcal{F} \).

In general, \( \kappa \)-plan \( \pi \) can be either a stationary (possibly partial) policy \( \pi : S \to O \), or a non-stationary policy \( \pi : S \times \mathbb{N} \to O \) that depends on the current state and the number of “failures so far.” As noted by Jensen et al. (2004), the latter can be captured as a stationary policy for a certain contingent planning task \( \Pi' \langle \mathcal{F}, \kappa \rangle \) that we refer to as \( \langle \mathcal{F}, \kappa \rangle \)-reformulation of \( \Pi \): Given a FT-planning task \( \langle \Pi = (P, O, s_0, G), \mathcal{F}, \kappa \rangle \), \( \Pi' \langle \mathcal{F}, \kappa \rangle \) is a contingent planning task over states \( S' \langle \mathcal{F}, \kappa \rangle = S \times \{0, \ldots, \kappa\} \), operators \( O \), initial state \( s_0 \langle \mathcal{F}, \kappa \rangle = (s_0, 0) \), and goal states \( S' \langle \mathcal{F}, \kappa \rangle = \bigcup_{i=0}^{\kappa} \{(s, i) \mid s \in S \} \). For each \( s \in S \), \( o \in O(s) \), and \( 0 \leq i \leq \kappa \), \( o \) is applicable in \( (s, i) \), and if applied, it changes the world to one of the states

\[
\text{Res}(s, i; o) = \bigcup_{\mathcal{F}(e) \geq 0} \{\text{Res}(s, i; e) \mid (s, i; e) \in \mathcal{F}(e)\}.
\]

where \( \text{Res}(s, i; e) = (\text{Res}(s; e), i + \mathcal{F}(e)) \).

It is not hard to verify that there is a bijection correspondence between plans for \( \Pi' \langle \mathcal{F}, \kappa \rangle \) and non-stationary \( \kappa \)-plans for \( \langle \Pi, \mathcal{F}, \kappa \rangle \), but the relation to stationary \( \kappa \)-plans for \( \langle \Pi, \mathcal{F}, \kappa \rangle \) is less immediate. Theorem 1 clarifies this matter. Let \( \pi \) be a (not necessarily a plan) policy for \( \Pi' \langle \mathcal{F}, \kappa \rangle \). The execution tree \( T_\pi(s, i) \) is the tree of possible executions of \( \pi \) starting at \( (s, i) \), with nodes corresponding to states of \( \Pi' \langle \mathcal{F}, \kappa \rangle \), edges corresponding to operator effects, and \( S' \langle \mathcal{F}, \kappa \rangle \subseteq S' \langle \mathcal{F}, \kappa \rangle \) denoting the set of internal nodes of \( T_\pi \). (Henceforth, \( T_\pi(s_0, 0) \) is referred to for short as \( T_\pi \).)

Theorem 1 Let \( \langle \Pi, \mathcal{F}, \kappa \rangle \) be a solvable FT-planning task. If \( \mathcal{F} \) is normative, then there exists a stationary \( \kappa \)-plan \( \pi \) for \( \langle \Pi, \mathcal{F}, \kappa \rangle \). In contrast, there exist solvable FT-planning tasks (with nonnormative exception models) for which there are no stationary \( \kappa \)-plans.

The proof of the second sub-claim is by example: Let \( \Pi \) be a contingent planning task over states \( S = \{s_0, s_1, s_2, s_3, s_4\} \), operators \( O = \{o_0, o_1, o_2\} \) as in Figure 1a, initial state \( s_0 \), and \( S_G = \{s_0\} \). Assume an exception model \( \mathcal{F} \) for \( \Pi \) as in the last column of Figure 1a. Figure 1b depicts the only contingent plan for the reformulation \( \Pi' \langle \mathcal{F}, 1 \rangle \), and the respective 1-plan for \( \langle \Pi, \mathcal{F}, 1 \rangle \) is not stationary: different actions, \( o_1 \) and \( o_2 \), are taken at state \( s_1 \) with 0 and 1 “exceptions so far,” respectively, and such a history-dependent choice of operator at \( s_1 \) is unavoidable.

Figure 1: Illustrations for the examples around Theorem 1.

Note that \( \mathcal{F} \) in the above example is not normative because neither of the effects of operator \( o_1 \) is primary. With normative exception models, the situation is indeed different. Let \( \langle \Pi, \mathcal{F}, \kappa \rangle \) be a solvable FT-planning task with a normative model \( \mathcal{F} \), and \( \pi \) be a contingent plan for \( \Pi' \langle \mathcal{F}, \kappa \rangle \).

If \( \pi(s, i) = \pi(s, j) \) for all pairs of reformulation states \( (s, i), (s, j) \in S' \langle \mathcal{F}, \kappa \rangle \), then we are done because the \( \kappa \)-plan for \( \langle \Pi, \mathcal{F}, \kappa \rangle \) corresponding to \( \pi \) is stationary. Otherwise, let \( (s, i), (s, j) \in S' \langle \mathcal{F}, \kappa \rangle \), \( i < j \) be a pair of reformulation states for which \( \pi(s, i) \neq \pi(s, j) \). The proof is accomplished by showing that \( \pi' \), obtained from \( \pi \) by replacing \( \pi(s, j) \) with \( \pi(s, i) \), is also a plan for \( \Pi' \langle \mathcal{F}, \kappa \rangle \), and thus we can always iteratively reduce a non-stationary \( \kappa \) to a stationary one.

Note that, by the construction used in the proof of Theorem 1, if \( \pi \) is a non-stationary \( \kappa \)-plan for a FT-planning task \( \langle \Pi, \mathcal{F}, \kappa \rangle \) with a normative model \( \mathcal{F} \), then \( \pi \) can be efficiently translated into a stationary \( \kappa \)-plan \( \pi' \) for \( \langle \Pi, \mathcal{F}, \kappa \rangle \). Likewise, for some of such pairs \( \pi \) and \( \pi' \), \( \pi' \) may turn out to be a strong cyclic contingent plan for \( \Pi \). For instance, let \( \Pi \) be a contingent planning task over states \( S = \{s_0, \ldots, s_4\} \), operators \( O = \{o_0, \ldots, o_3\} \), initial state \( s_0 \), and \( G = s_4 \). The operators are defined as in the table in Figure 1c, and the exception model \( \mathcal{F} \) associated with \( \Pi \) is given in the last column of that table. Figure 1d depicts a contingent plan \( \pi \) for the reformulation \( \Pi' \langle \mathcal{F}, \kappa \rangle \), and the respective 1-plan for \( \langle \Pi, \mathcal{F}, 1 \rangle \) is not stationary: different actions, \( o_0 \) and \( o_3 \), are taken at state \( s_0 \) with 0 and 1 “exceptions so far,” respectively. However, if we modify \( \pi \) as in the proof of Theorem 1, the resulting plan \( \pi' \) for \( \Pi' \langle \mathcal{F}, \kappa \rangle \) will induce a strong cyclic contingent plan \( \pi' \langle \mathcal{F}, \kappa \rangle \) for \( \Pi \).

**Complexity and Compilation**

Two decision problems are of interest in the context of FT-planning: Let \( \Pi \) be a contingent planning task, and \( \mathcal{F} \) be an \( \alpha \)-primary model for \( \Pi \).

**FT-PLAN-\( \alpha \)-\( \kappa \):** Does \( \Pi \) have a \( \kappa \)-plan?
POLY-FT-PLAN-\(\alpha\)-\(\kappa\): Does \(\Pi\) have a \(\kappa\)-plan such that all its \(\kappa\)-admissible executions reach the goal after a polynomial number of steps?

At first view, the effective difference between FT-PLAN-\(\alpha\)-\(\kappa\) and contingent planning is not clear. In general, for sufficiently large values of \(\alpha\) (e.g., \(\alpha = |S|\)), contingent planning can trivially be reduced to FT-PLAN-\(\alpha\)-\(\kappa\) for any \(\kappa\), and hence the latter decision problem is \text{EXPTIME}-hard. In fact, FT-PLAN-\(\alpha\)-\(\kappa\) can be polynomially reduced to FT-PLAN-2-\(\kappa\) by simulating each operator with \(\alpha\) primary effects by a “ladder” of \(\log \alpha\) operators, each with at most two primary effects. Hence, even FT-PLAN-2-\(\kappa\) is \text{EXPTIME}-hard. However, while the definition of exception models is rather general, the specific settings that brought us to this investigation correspond to the normative 1-primary exception models considered by Jensen, Veloso, and Bryant (2004). In what comes next, we focus on that fragment of FT-planning.

**FT-planning with 1-primary models**

Unlike FT-PLAN-2-\(\kappa\), FT-PLAN-1-\(\kappa\) nicely generalizes classical planning, which simply corresponds to first associating the contingent planning task with a normative 1-primary model, and then adopting the extreme optimism that no failures will occur along the purported plan to the goal. In other words, the decision version of classical planning is precisely the FT-planning class FT-PLAN-1-0. At the same time, FT-PLAN-1-1 already goes way beyond classical planning. While plans for FT-PLAN-1-1 are restricted to at most one operator failure per possible plan execution, these failures are bounded neither to specific operators nor to specific stages of the purported plan. Hence, while plans for FT-PLAN-1-0 are linear sequences of actions, plans for FT-PLAN-1-\(\kappa\) with \(\kappa > 0\) are tree-structured, and may actually exhibit substantial branching: unlike in (\text{EXPSPACE-complete}) planning under the \(k\)-branching assumption (Bonet 2010), plans for FT-PLAN-1-\(\kappa\) may have to always interleave between acting and branching, even for \(\kappa = 1\).

For example, suppose that a robot should move from \(x_1\) to \(x_2\) on the map depicted in Figure 2(a). Movements on the segments \((x_1, x_2)\) and \((x_1, x_3)\) are considered safe and thus are modeled by deterministic operators. Movements \(\text{move}(x_i, x_j)\) on the other three segments are modeled by non-deterministic operators with three possible effects: \(\text{move}(x_i, x_j)\) typically brings the robot to \(x_j\), with no side effects, but it may also bring the robot to \(x_j\) with a flat tire, or keep it at \(x_i\) for the same reason. Initially the robot has no flats, but also no spare tires. A single spare tire can be picked up at each of the two intermediate locations \(x_2\) and \(x_3\). Figures 2(b-d) depict stationary 0-plan, 1-plan, and 2-plan for the respective FT-planning tasks, under

\[\text{Figure 2: \(\kappa\)-plans for an inline example}\]

a normative 1-primary exception model that maps the primary effects of all actions to 0, and the exceptional effects of the non-deterministic move actions to 1. In the triplet denotation \([x, y, z]\) of the states, \(x\) is the robot’s location, \(y \in \{\text{ok}, \text{F}\}\) is the status of the tire, and \(z\) is the number of spare tires in the robot’s possession. State representation in this problem also addresses the availability of the spare tires at \(x_2\) and \(x_3\), but we omit this information in the figure for brevity. The dashed arrows depict possible effects of the actions that the agent ignores at planning. The 0-plan is a classical plan, and it is as simple as the example itself. The 1-plan is already more involved: to guarantee reaching \(x_5\) under a possibility of a single fault, the robot picks up a spare tire at \(x_3\). In turn, the 2-plan in Figure 2d prescribes that the robot first collect both spare tires at \(x_2\) and \(x_3\), and only then start moving towards \(x_5\), replacing flat tires on the way, if needed.

Still, despite the structural complexity of \(\kappa\)-plans, the \text{EXPTIME}-hardness proof for FT-PLAN-2-\(\kappa\) does not carry over to FT-PLAN-1-\(\kappa\), and for a good reason: Theorem 2 below shows that FT-PLAN-1-\(\kappa\) is in PSPACE, that is, worst-case not harder than classical planning. Moreover, Theorem 3 then shows even closer resemblance between these two formalisms, namely that POLY-FT-PLAN-1-\(\kappa\) is in NP.

**Theorem 2** FT-PLAN-1-\(\kappa\) is in PSPACE.

A non-deterministic algorithm (that can also be compiled to a Turing machine) for deciding whether there is a \(\kappa\)-plan
Algorithm BO-PLAN-1-κ(Π, F)

main
Plan-FT(s0, κ)
accept

procedure Plan-FT(s, k)
steps ← 0
while steps < 2n
if s = G then return
choose operator o s.t. s = pre(o)
for e ∈ eff(o)
do
if F(e) > k
then continue // under assumed κ, e cannot happen here
else if F(e) > 0
then Plan-FT( Res[s; e], k − F(e) )
else
steps ← steps + 1
s ← Res[s; e]

reject

Figure 3: PSPACE algorithm for deciding FT-PLAN-1-κ.

for an FT-planning task (Π, F, κ) with 1-primary F and |Π| = n is depicted in Figure 3. The respective Turing machine is in PSPACE because, at any point, there are at most κ open calls to the Plan-FT procedure, each storing a single state in n bits and a single counter steps in n bits. Finally, since it is in PSPACE, FT-PLAN-1-κ is also PSPACE-complete by the PSPACE-hardness of classical planning under the description language we use, and equivalence of the latter to FT-PLAN-1-0.

Theorem 3 POLY-FT-PLAN-1-κ is in NP.

The proof is by showing that, if (Π, F, κ) has a κ-plan π′ such that all its κ-admissible executions reach the goal after O(nκ) steps, then there is a κ-admiss pl π with the same property such that ||π|| = O(b(k+1)κ(n+2)), where b = maxs∈O |eff(o)|. Since b = O(|Π||π|) and both c and κ are O(1), the rest stems from the standard guess-and-verify argument of NP-membership.

First, since κ-plans guarantee goal reachability only along executions with κ or fewer exceptions, let π follow π′ on states reachable by the κ-admissible executions of π′, and make a random operator choice everywhere else. Thus, only the κ-admissible executions of π should be represented. Second, as we upper-bound the description size of π, for simplicity we assume (i) extensive, tree-structured representation of π, and (b) that the range of F is {0, 1}. For 0 ≤ i ≤ κ, let f(π, i) be the number of i-admissible executions of π that are not (i − 1)-admissible. Clearly, \( \sum_{i=0}^{κ} f(π, i) \) is the overall number of κ-admissible executions, and thus \( ||π|| = O(nκ \sum_{i=0}^{κ} f(π, i)) \). Since F is 1-primary, there is at most one 0-admissible execution of π per possible initial state, and thus \( f(π, 0) = 1 \). Recursively, due to the same argument of F being 1-primary, each of the O(nκ) operator instances along that single 0-admissible execution may branch into b 1-admissible executions of π. However, these are the only possible sources of 1-admissible executions. Thus, \( f(π, 1) = O(f(π, 0) \cdot bn^κ) = O(bn^κ) \), and in general, \( f(π, i) = O((bn^κ)^i) \). Hence, \( ||π|| = O(n^κ \cdot (bn^κ)^{κ+1}) = O(b(k+1)n^κ+2) \).

Note that, while κ = O(1) should suffice for most interests in practice, the PSPACE-membership result of Theorem 2 holds for κ = O(poly(|Π|)). This is not so, however, with the NP-membership result of Theorem 3, which relies upon κ = O(1).

Compilation to Classical Planning

Theorems 2 and 3 put FT-planning under 1-primary exception models rather close to classical planning. On the one hand, that suggests that classical planning machinery can possibly be adapted to solve such FT-planning tasks. On the other hand, the non-linearity of κ-plans under 1-primary models seems to complicate applying classical planning algorithms to FT-planning.

We now show that FT-planning under 1-primary models can be efficiently compiled into classical planning, at least as long as the number of non-deterministic effects per operator is bounded by a constant. The compilation is to STRIPS with negative preconditions and conditional effects. In this formalism, a planning task is given by a quadruple \( (P, O, s_0, G) \), with \( P, s_0 \), and \( G \) being as in our formalism for contingent planning. Operators \( o \in O \) are pairs (pre(o), eff(o)) where the precondition pre(o) is a subset of literals over \( P \), and eff(o) is a set of conditional effects. A conditional effect e is a triplet (con(e), add(e), del(e)) of condition, add, and delete lists, respectively, where con(e) is a subset of literals over \( P \), while add(e) and del(e) are subsets of propositions. Operator o is applicable in state s iff s |= pre(o). If o ∈ O(s) is applied in s, it deterministically changes the world to state res[s; o] = \( \bigcup_{e \in \text{eff}(o), s \models \text{con}(e)} (s \setminus \text{del}(e)) \cup \text{add}(e) \).

We start with a compilation of a simple fragment of FT-PLAN-1-κ, corresponding to FT-planning tasks \( (Π = (P, O, s_0, G), F, κ) \) such that (i) each operator of Π has at most two effects, and (ii) F is a normative 1-primary exception model for Π such that, for each o ∈ O(s), if eff(o) = \{e0\}, then F(e0) = 0, and if eff(o) = \{e0, e1\}, then F(e0) = 0 and F(e1) = 1. We begin with an example that illustrates the basic idea behind this compilation. Let κ = 2, and let Figure 4a depict an irreducible contingent plan π for Π(F, κ) such that the arcs correspond to the operator effects and are labeled with the respective values of F, and double-frame states are the goal states.

Note that, in Figure 4a, the states and operator instances in \( π \) are numbered consistently with a DFS traversal of the execution tree \( T_π \). Therefore, the operator sequence \( o_1, \ldots, o_7 \) induces a sequence of policies \( π_0, \ldots, π_7 \) for Π(F, κ) such that \( π_0 \) is an empty policy, \( π_7 = π \), and each πi extends πi−1 with mapping a single leaf of \( T_{π_i−1} \) to operator \( o_i \). An important property of this sequence of policies \( π_0, \ldots, π_7 \) is that, for 0 ≤ j ≤ 2, each πi induces at most one execution sequence of πi\( \perp \) with F(ρ) = j that does not achieve the goal within \( T_π \).

The latter is emphasized by the tabular representation of this sequence of policies in Figure 4b. The columns in the table capture certain subsets \( σ_0, \ldots, σ_7 \) of leaves of \( T_{π_0}, \ldots, T_{π_7} \), that is, of the end-states of \( π_0, \ldots, π_7 \), respectively. For
Figure 4: A plan $\pi$ for $\Pi^{(F,\kappa)}$, and the end-state set representation of the induced sequence of $\pi$’s sub-policies.

$0 \leq j \leq 2$, each $\sigma_i$ contains at most one state with “$j$ failures so far,” denoted $\sigma_i(j)$, with $\sigma_i(j) = \perp$ denoting that $\sigma_i$ does not contain a state for $j$. For $i = 0$, $\sigma_0(0) = (s_0, 0)$, and $\sigma_0(1) = \sigma_0(2) = \perp$. For $i > 0$, if $T_\pi$ has a (unique) leaf $(s, j)$ such that $s \notin S_G$, then $\sigma_i(j) = (s, j)$, and otherwise, $\sigma_i(j) = \sigma_{i-1}(j)$. Asterisks in the table are by the goal states, and note that the last set $\sigma_7$ is the first in the sequence to contain only goal states.

It turns out that any irreducible $\kappa$-plan for any FT-planning task from the fragment in question induces such a DFS-ordered sequence of sub-policies with “at most one non-goal leaf with $j$ failures so far.” It is precisely this property that provides a basis for our compilation of $\langle \Pi = \langle P, O, s_0, G, F, \kappa \rangle \rangle$ to an equivalent classical planning task $\Pi’ = \langle P’, O’, s_0’, G’ \rangle$. For now, we postpone the formal statement and proof of this property, and formulate the actual compilation. After that, in Lemma 5, we formally state the task properties underlying the compilation, and use this lemma to specify compilation for a wider class of FT-planning tasks.

**Propositions.** For clarity, we use letters $s$ and $\sigma$ to denote states of $\Pi$ and $\Pi’$, respectively. The subsets $s_0, \ldots, s_7$ of $S^{(F,\kappa)}$ in Figure 4b hint at the state space structure of $\Pi’$: Each reachable state $\sigma$ of $\Pi’$ represents a partial policy $\pi_\sigma$ for $\Pi^{(F,\kappa)}$ that corresponds to a concrete stage of a certain DFS traversal of the purported tree-structured plan for $\Pi^{(F,\kappa)}$. To support that, $P’$ contains $\kappa + 1$ replicas $P_0, \ldots, P_\kappa$ of $P$, as well as a set of auxiliary propositions $\langle \text{open}_i \rangle_{i=0}^\kappa$. The interpretation of $\text{open}_i \in \sigma$ is that “the policy for $\Pi^{(F,\kappa)}$ represented by $\sigma$ induces an $i$-admissible execution that does not achieve the goal, and the end-state of that execution is captured by the values of $P_i$ in $\sigma$.” In what follows, by $\sigma/P_i$ we denote the valuation provided by $\sigma$ to propositions $P_i$. Likewise, if $\phi$ is a set of literals either over $P$ or over one of the proposition sets $P_0, \ldots, P_\kappa$ of $\Pi’$, then, for $0 \leq i \leq \kappa$, $\phi_i$ is a set of literals over $P_i$, obtained from $\phi$ by replacing all the propositional symbols with their counterparts in $P_i$. For example, $\langle \{p, \neg q\} \rangle_i = \{p_i, \neg q_i\}$, and $\langle \{p_i, \neg q_i\} \rangle_i = \{p_j, \neg q_j\}$.

**Operators.** $O’$ contains $\kappa + 1$ sets of operators $O_0, \ldots, O_\kappa$, with $O_i = \{o_i \mid o \in O\}$. For each $o \in O$, the precon of the operator $o_i$ is 

$$\text{pre}(o_i) = |\text{pre}(o_i)|_i \cup \{\text{open}_i\} \cup \bigcup_{j=i+1}^\kappa \{\neg\text{open}_j\}.$$ 

In other words, if state $\sigma$ of $\Pi’$ represents a policy for $\Pi^{(F,\kappa)}$ such that some admissible execution of that policy do not reach the goal, then the planner is forced to extend the $i$-admissible such execution with the highest index $i$. If either $o$ is deterministic or $i = \kappa$, then $\text{eff}(o_i) = \text{Res}[\sigma/P_i; [e_0]]$. Otherwise, if $\text{eff}(o) = \{e_0, e_1\}$ and $i < \kappa$, then 

$$\text{eff}(o_i) = \text{Res}[\sigma/P_i; [e_0]] \cup \text{Res}[\sigma/P_i; [e_1]]_{i+1} \cup \text{open}_{i+1}.$$ 

In the formalism of our choice, such operator effects are captured as follows. If $\text{eff}(o) = \{e_0\}$ or $i = \kappa$, then $\text{eff}(o_i)$ contains a single unconditional effect:

$$\text{eff}(o_i) = \{ (\emptyset, [\text{add}(e_0)]_i, [\text{del}(e_0)]_i) \}.$$ 

Otherwise, if $\text{eff}(o) = \{e_0, e_1\}$ and $i < \kappa$, then 

$$\text{eff}(o_i) = \Phi_i \cup \left\{ (\emptyset, [\text{add}(e_0)]_i, [\text{del}(e_0)]_i), (\emptyset, [\text{add}(e_1)]_{i+1}, [\text{del}(e_1)]_{i+1}) \right\},$$

where 

$$\Phi_i = \bigcup_{p \notin [\text{add}(e_0)]_i, [\text{del}(e_1)]_i} \left\{ (\{p_i\}, \{p_{i+1}\}, \emptyset), (\emptyset, \{p_i\}, \{p_{i+1}\}) \right\}$$

compactly encodes the situation calculus frame axioms between the “current situation with $i$ failures so far” and the “next situation with $i + 1$ failures so far.” In addition, $O’$ contains a set of auxiliary “goal-achieving” operators $\{o_i^{\ast}\}_{i=0}^\kappa$ with 

$$\text{pre}(o_i^{\ast}) = [G_i] \cup \{\text{open}_i\} \cup \bigcup_{j=i+1}^\kappa \{\neg\text{open}_j\},$$

$$\text{eff}(o_i^{\ast}) = \{ (\emptyset, \emptyset, [\text{open}_i]) \}.$$ 

**Initial state and goal.** The $P_0$-part of the initial state $\sigma_0^{\ast}$ captures the sole initial state of $\Pi$, and for $i > 0$, the $P_i$-parts of the initial state are actually not important, and can be set arbitrarily. The auxiliary variable $\text{open}_0$ is initially set to true, the rest of the variables $\text{open}_i$ are initially set to false, and the goal of $\Pi’$ is to negate $\text{open}_0$. That is, $\sigma_0^{\ast} = [s_0]_0 \cup \{\text{open}_0\}$ and $G’ = \{\neg\text{open}_0\}$.

For an illustration, consider a small and simplified variant of our running example in which there are only two locations, $x$ and $\text{not}(x)$, the robot and a single spare tire are initially at $x$, and the goal is for the robot to be at $\text{not}(x)$. Movement of the robot either succeeds (which is the primary effect of that operator), or fails, with the robot staying at the original location with a flat tire. This planning task is encoded using propositions $P = \{x, \text{not flat}, \text{spare}\}$ by the operators as in the Table 1a, initial state $s_0 = \{x, \text{spare}, \text{not flat}\}$, and goal $G =$
\{x\}. The compilation \( \Pi' = (P', O', \sigma_0', G') \) of the FT-planning task \((\Pi, F, 1)\) is defined over propositions \( P' = \bigcup_{i \in \{0, 1\}} \{x, noflat, spare, open\} \), and operators as in Table 1b. The initial state is \( \sigma_0' = \{x_0, spare_0, noflat_0, open_0\} \), and the goal is \( G' = \{\neg open_0\} \). It is not hard to verify that \( \pi = \{move_0, fix_1, move_1, \sigma_0', \sigma_0''\} \) is the only plan for the classical planning task \( \Pi' \), and that the respective contingent plan for \( \Pi'_{(F, 1)} \) can be decoded from \( \pi \) in linear time.

In the spirit of this example, we have generated a set of tasks in which a robot should move from the bottom-left to the top-right corner of a 4-connected grid, in which some of the edges are "safe," with moves along them being deterministic, while other edges are "unsafe," with moves on them either succeeding (which is the primary, expected effect of that operator) or resulting in the robot getting a flat and staying where it was. A limited number of spare tires are available on some nodes of the grid. The six sets of five tasks each correspond to 5 × 5 and 7 × 7 grids, with each edge of the grid being independently marked as safe with probability \( p \in \{0.1, 0.2, 0.5\} \), and 10 spare tires, independently positioned on the grid nodes at random.

The runtimes of different approaches on these tasks are depicted in Table 2. The three approaches we examined were (col. 2) contingent planning with Contingent-FF (Hoffmann and Brafman 2005); (col. 3-6) FT-planning with Contingent-FF over \((F, \kappa)\)-reformulations, \( \kappa \in \{0, 1, 2, 4\} \), and (col. 7-10) FT-planning with Fast Downward’s GBFS with FF heuristic over the classical planning reductions of the \((F, \kappa)\)-reformulations. Each task/planner was given a 10 minute time limit; cases in which the planner neither solved the task nor proved it unsolvable within the time bound are marked with '-'.

As Table 2 shows, all but one of these tasks were proven by Contingent-FF to have no strong contingent plans (and cyclic contingent plans are also of no help in this domain), while all (effectively classical) FT-planning tasks with \( \kappa = 0 \) were easily solved by both Contingent-FF and Fast Downward.

### Table 1: Operators from the compilation example.

<table>
<thead>
<tr>
<th>( o )</th>
<th>( pre )</th>
<th>( eff )</th>
<th>( F' )</th>
</tr>
</thead>
<tbody>
<tr>
<td>move</td>
<td>( {x, noflat} )</td>
<td>( e_0 = \emptyset, {x} )</td>
<td>( F(e_0) = 0 )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( e_1 = \emptyset, {noflat} )</td>
<td>( F(e_1) = 1 )</td>
</tr>
<tr>
<td>fix</td>
<td>( {x, spare} )</td>
<td>( e_0 = \emptyset, {spar} )</td>
<td>( F(e_0) = 0 )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( o )</th>
<th>( pre )</th>
<th>( eff )</th>
</tr>
</thead>
<tbody>
<tr>
<td>move_0</td>
<td>( {x_0, noflat_0, open_0} )</td>
<td>( \emptyset, \emptyset, {x_0} )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \emptyset, {open_1}, \emptyset )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \emptyset, \emptyset, {noflat_0} )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( {x_0}, {x_1}, \emptyset )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( {\neg x_0}, \emptyset, {x_1} )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( {\neg spare_0}, {spare_1}, \emptyset )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( {\neg spare_0}, \emptyset, {spar_1} )</td>
</tr>
<tr>
<td>fix_0</td>
<td>( {x_0, spare_0, open_0} )</td>
<td>( \emptyset, {noflat_0}, {spare_0} )</td>
</tr>
<tr>
<td>fix_1</td>
<td>( {x_1, spare_1, open_1} )</td>
<td>( \emptyset, {noflat_1}, {spare_1} )</td>
</tr>
<tr>
<td>( o_0' )</td>
<td>( \neg x_0, open_0, \neg open_1 )</td>
<td>( \emptyset, {open_0} )</td>
</tr>
<tr>
<td>( o_1' )</td>
<td>( \neg x_1, open_1, \neg open_1 )</td>
<td>( \emptyset, {open_1} )</td>
</tr>
</tbody>
</table>

### Table 2: Planner runtimes on different formulations of FT-planning tasks in the spirit of our example.

<table>
<thead>
<tr>
<th>Task type</th>
<th>Contingent-FF ((\Pi_{(F, \kappa)}^c))</th>
<th>Fast Downward ((F))</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \kappa \leq 0 )</td>
<td>0:08</td>
<td>0:03</td>
</tr>
<tr>
<td>( \kappa = 0 )</td>
<td>0:08</td>
<td>0:02</td>
</tr>
<tr>
<td>( \kappa &gt; 0 )</td>
<td>0:08</td>
<td>0:04</td>
</tr>
</tbody>
</table>

ward. For us, of course, the interesting part was in between these two extremes, and both Contingent-FF and Fast Downward found non-trivial \( \kappa \)-plans for numerous tasks here. In terms of performance, compiling the contingent \((F, \kappa)\)-reformulations to classical planning strictly dominated solving the former directly, in terms of the coverage of both solvable and unsolvable FT-planning tasks. In sum, the direction of compiling FT-planning tasks to classical planning appears promising, and clearly deserves further investigation.

In Lemma 5 below, we now formalize the properties of FT-PLAN-1-\( \kappa \) that are exploited by the compilation of its fragment above. In particular, this lemma allows for extending this compilation scheme to arbitrary fixed bounds on the number of non-deterministic effects per operator, as well as to arbitrary normative 1-primary exception models.

**Lemma 4** Let \( \Pi = (P, O, s_0, G, F, \kappa) \) be an FT-planning task with a 1-primary model \( F \), and \( \text{max}_{o \in O} \text{eff}(o) = b \). If \( \pi \) is an irreducible contingent plan for \( \Pi_{(F, \kappa)} \), then there exists a set of policies \( \pi_0, ..., \pi_m \) over \( S(F, \kappa) \) such that

1. \( \pi_0 \) is an empty policy, \( \pi_m = \pi \), and each \( \pi_i \) extends \( \pi_{i-1} \) by prescribing an action for a single additional state of \( \Pi_{(F, \kappa)} \) such that the execution tree \( T_{\pi_{i-1}} \) is a proper sub-tree of \( T_\pi \), and
2. for \( 0 \leq i \leq m \), \( 0 \leq j \leq \kappa \), \( \pi_i \) induces at most \( b \) executions \( \rho \) that do not achieve the goal within \( T_\pi \) and have \( F(\rho) = j \).

The proof is as follows. Let \( \pi \) be an irreducible contingent plan for the \((F, \kappa)\)-reformulation of \( \Pi_{(F, \kappa)} \) as in the claim, and let \( \{s_1, k_1\}, ..., \{s_m, k_m\} \) be a relabeling of the nodes \( S_\pi \) consistently with the order in which
they are expanded by a depth-first traversal of \( T_\pi \), with the “depth” of a node \((s, k)\) being given by \( k \). Given that, let a sequence of policies \( M = \pi_0, \ldots, \pi_m \) be defined as:

\[
\pi_i(s_j, k_j) = \begin{cases} 
\pi_i(s_j, k_j), & j \leq i \\
\text{undefined}, & j > i
\end{cases}
\]

It is immediate that \( M \) satisfies condition (1) of the lemma, and so what remains to be shown is condition (2). The proof is by induction on \( i \). For \( i = 0 \), the condition is trivially satisfied since \( \pi_0 \) is empty. Assuming that the condition is satisfied for \( i - 1 \leq 0 \), the proof for \( i \) is as follows. By the DFS construction of \( M \), we have \( \sigma_\pi(s_j, k_j) = \sigma_\pi(s_{i-1}, (s_i, k_i)) \), where \( (s_i, k_i) \) is a non-goal leaf node in \( T_{\pi_{i-1}} \). Furthermore, for all other non-goal leaf nodes \((s_j, k_j)\) of \( T_{\pi_{i-1}} \), it holds that \( k_j \leq k_i \), or otherwise DFS would expand \((s_j, k_j)\) prior to \((s_i, k_i)\). Given that, consider the extension of \( T_{\pi_{i-1}} \) to \( T_{\pi_{i}} \) by \( \pi(s_i, k_i) \).

By the definition of exception models, for each \( k < k_i \), the number of executions \( \rho \) of \( \pi_i \) that do not achieve the goal within \( T_{\pi_{i}} \) and have \( \mathcal{F}(\rho) = k \) is the same as for \( \pi_{i-1} \), and this because the number of “exceptions so far” cannot decrease with the progress of the execution. For \( k = k_i \), since \( \mathcal{F} \) is 1-primary, \( \pi_i \) replaces a single execution \( \rho \) of \( \pi_{i-1} \) that does not achieve the goal within \( T_{\pi_{i-1}} \) and has \( \mathcal{F}(\rho) = k_i \), with at most one such execution, namely the one that extends \( \rho \) with the sole primary effect of \( \pi(s_i, k_i) \). Finally, for all \( k > k_i \), there are no executions of \( \pi_{i-1} \) that do not achieve the goal within \( T_{\pi_{i-1}} \) and have \( \mathcal{F}(\rho) = k \), and thus there are at most \( b \) such executions of \( \pi_i \).

Given an FT-planning task \( \langle \Pi = \langle P, O, s_0, G, F, \kappa \rangle \rangle \) with a normative 1-primary model \( \mathcal{F} \) and \( \max_{o \in O} \text{eff}(o) = O(1) \), a polynomial-time, sound, and complete compilation of \( \langle \Pi, \mathcal{F}, \kappa \rangle \) to a classical planning task \( \Pi' = \langle P', O', \sigma_0', G' \rangle \) is specified below. For ease of presentation, for each operator \( o \in O(s) \), if \( \text{eff}(o) = \{e_0, e_1, \ldots, e_{b-1}\} \), then \( \mathcal{F}(e_0) = 0 \). The set of propositions \( P' \) contains \( \kappa(b-1)+1 \) replicas of propositions \( P \), as well as \( \kappa(b-1)+1 \) auxiliary propositions \( P_i \), denoted as:

\[
P' = P_{0,0} \cup \{\text{open}_{0,0}\} \cup \bigcup_{1 \leq x \leq \kappa} P_{i,j} \cup \{\text{open}_{i,j}\}.
\]

The set of operators \( O' \) contains \( \kappa(b-1)+1 \) sets of operators \( O_{0,0}, O_{1,0}, \ldots, O_{b-1,0} \), \( O_{0,1}, O_{1,1}, \ldots, O_{b-1,1} \), \ldots, \( O_{0,b-1}, O_{1,b-1}, \ldots, O_{b-1,b-1} \), with \( O_{i,j} = \{o_{i,j} | o \in O\} \). For each \( o \in O \), the precondition of the operator \( o_{i,j} \) is:

\[
\text{pre}(o_{i,j}) = \{\text{pre}(o)_{i,j} \} \cup \{\text{open}_{i,j}\} \cup \bigcup_{x=j+1}^{b-1} \{\neg \text{open}_{i,x}\}.
\]

If \( \text{eff}(o) = \{e_0, e_1, \ldots, e_{b-1}\} \), then:

\[
\text{eff}_{i,j} = \{\langle \emptyset, \text{add}(e_0)_{i,j} \cup \text{del}(e_0)_{i,j} \rangle \} \cup \bigcup_{1 \leq x \leq y \leq \kappa} \{\langle \emptyset, \text{add}(e_i)_{i,j} \cup \text{del}(e_i)_{i,j} \rangle \} \cup \Phi_{i,j,x}
\]
References


Göbelbecker, M.; Gretton, C.; and Dearden, R. 2011. A switching planner for combined task and observation planning. In AAAI.


Kissmann, P., and Edelkamp, S. 2012. Improving cost-optimal domain-independent symbolic planning. In AAAI.


Taig, R., and Brafman, R. I. 2013. Compiling conformant probabilistic planning into classical planning. In ICAPS.

