Planning with Incomplete Information in Quantified Answer Set Programming

Jorge Fandinno[2,3] François Laferrière[3] Javier Romero[3] Torsten Schaub[3] Tran Cao Son[1]
[1]*New Mexico State University, USA* [2]*Omaha State University, USA* [3]*University of Potsdam, Germany*

submitted 1 January 2003; revised 1 January 2003; accepted 1 January 2003

Abstract

We present a general approach to planning with incomplete information in Answer Set Programming (ASP). More precisely, we consider the problems of conformant and conditional planning with sensing actions and assumptions. We represent planning problems using a simple formalism where logic programs describe the transition function between states, the initial states and the goal states. For solving planning problems, we use Quantified Answer Set Programming (QASP), an extension of ASP with existential and universal quantifiers over atoms that is analogous to Quantified Boolean Formulas (QBFs). We define the language of quantified logic programs and use it to represent the solutions to different variants of conformant and conditional planning. On the practical side, we present a translation-based QASP solver that converts quantified logic programs into QBFs and then executes a QBF solver, and we evaluate experimentally the approach on conformant and conditional planning benchmarks. Under consideration for acceptance in TPLP.

1 Introduction

We propose a general and uniform framework for planning in Answer Set Programming (ASP; Lifschitz 2002). Apart from classical planning, referring to planning with deterministic actions and complete initial states, our focus lies on conformant and conditional planning. While the former extends the classical setting by incomplete initial states, the latter adds sensing actions and conditional plans. Moreover, we allow for making assumptions to counterbalance missing information.

To illustrate this, let us consider the following example. There is a cleaning robot in a corridor going through adjacent rooms that may be occupied by people. The robot can go to the next room and can sweep its current room to clean it, but it should not sweep a room if it is occupied. We assume that nothing changes if the robot tries to go further than the last room, or if it sweeps a room that is already clean. The goal of the robot is to clean all rooms that are not occupied by people. We present a solution for any number of rooms, but in our examples we consider only two.

Classical planning. Consider an initial situation where the robot is in the first room, only the first room is clean, and no room is occupied. In this case, the classical planning problem is to find a plan that, applied to *the* initial situation, achieves the goal. The plan, where the robot goes to the second room and then sweeps it, solves this problem.

Conformant planning. Consider now that the robot initially does not know whether the rooms are clean or not. There are four possible initial situations, depending on the state of cleanliness of the two rooms. In this case, the conformant planning problem is to find a plan that, applied to *all* possible initial situations, achieves the goal. The plan, where the robot sweeps, goes to the second room and sweeps it, solves that problem.

So far rooms were unoccupied. Now consider that initially the robot knows that exactly one of the rooms is occupied, but does not know which. Combining the previous four options about cleanliness with the two new options about occupancy, there are eight possible initial situations. It turns out that there is no conformant plan for this problem. The robot would have to leave the occupied room as it is and sweep the other, but there is no way of doing that without knowing which is the occupied room.

Assumption-based planning. At this point, the robot can try to make assumptions about the unknown initial situation, and find a plan that at least works under these assumptions, hoping that they will indeed hold in reality. In this case, a conformant planning problem with assumptions is to find a plan and a set of assumptions such that the assumptions hold in some possible initial situation, and the plan, applied to *all* possible initial situations satisfying the assumptions, achieves the goal. Assuming that room one is occupied, the plan where the robot goes to room two and then sweeps it, solves the problem. Another solution is to assume that the second room is occupied and simply sweep the first room.

Conditional planning. The robot has a safer approach, if it can observe the occupancy of a room and prepare a different subplan for each possible observation. This is similar to conformant planning, but now plans have actions to observe the world and different subplans for different observations. In our example, there is a conditional plan where the robot first observes if the first room is occupied, and if so, goes to the second room and sweeps it, otherwise it simply sweeps the first room. The robot could also make some assumptions about the initial situation, but this is not needed in our example.

Unfortunately, the expressiveness of regular ASP is insufficient to capture this variety of planning problems. While bounded classical and conformant planning are still expressible since their corresponding decision problems are still at the first and second level of the polynomial hierarchy, bounded conditional planning is PSPACE-complete (Turner 2002).

To match this level of complexity, we introduce a quantified extension of ASP, called Quantified Answer Set Programming (QASP), in analogy to Quantified Boolean Formulas (QBFs). More precisely, we start by capturing the various planning problems within a simple uniform framework centered on the concept of transition functions, mainly by retracing the work of Son et al. (2005; 2007; 2011). The core of this framework consists of a general yet simplified fragment of logic programs that aims at representing transition systems, similar to action languages (Gelfond and Lifschitz 1998) and (present-centered) temporal logic programs (Cabalar et al. 2018), We then extend the basic setting of ASP with quantifiers and define quantified logic programs, in analogy to QBFs. Although we apply QASP to planning problems, it is of general nature. This is just the same with its implementation, obtained via a translation of quantified logic programs to QBFs. This allows us to represent the above spectrum of planning problems by quantified logic programs and to compute their solutions with our QASP solver. Interestingly, the core planning problems are described by means of the aforementioned simple language fragment, while the actual type of planning problem is more or less expressed via quantification. Finally, we empirically evaluate our solver on conformant and conditional planning benchmarks.

2 Background

We consider normal logic programs over a set $\mathcal{A}$ of atoms with choice rules and integrity constraints. A rule r has the form $H \leftarrow B$ where B is a set of literals, and H is either an atom p, and we call r a normal rule, or $\{p\}$ for some atom p, making r a choice rule, or $\bot$, so that r an integrity constraint. We usually drop braces from rule bodies B, and also use l, B instead of $\{l\} \cup B$ for a literal l. We also abuse notation and identify sets of atoms X with sets of facts $\{p \leftarrow \ \mid p \in X\}$. A (normal) logic program is a set of (normal) rules. As usual, rules with variables are viewed as shorthands for the set of their ground instances. We explain further practical extensions of ASP, like conditional literals and cardinality constraints, in passing. Semantically, we identify a body B with the conjunction of its literals, the head of a choice rule $\{p\}$ with the disjunction $p \vee \neg p$, a rule $H \leftarrow B$ with the implication $B \rightarrow H$, and a program with the conjunction of its rules. A set of atoms $X \subseteq \mathcal{A}$ is a stable model of a logic program P if it is a subset-minimal model of the formula that results from replacing in P any literal by $\bot$ if it is not satisfied by X. We let $SM(P)$ stand for the set of stable models of P.

The dependency graph of a logic program P has nodes $\mathcal{A}$, an edge $p \xrightarrow{+} q$ if there is a rule whose head is either q or $\{q\}$ and whose body contains p, and an edge $p \xrightarrow{-} q$ if there is a rule with head q or $\{q\}$, and $\neg p$ in its body. A logic program is stratified if its dependency graph has no cycle involving a negative edge ($\xrightarrow{-}$). Note that stratified normal programs have exactly one stable model, unlike more general stratified programs.

ASP rests on a Generate-define-test (GDT) methodology (Lifschitz 2002). Accordingly, we say that a logic program P is in GDT form if it is stratified and all choice rules in P are of form $\{p\} \leftarrow$ such that p does not occur in the head of any other rule in P. In fact, GDT programs constitute a normal form because every logic program can be translated into GDT form by using auxiliary atoms (Niemelä 2008; Fandinno et al. 2020).

Quantified Boolean formulas (QBFs; Giunchiglia et al. 2009) extend propositional formulas by existential ($\exists$) and universal ($\forall$) quantification over atoms. We consider QBFs over $\mathcal{A}$ of the form

$$Q_0 X_0 \dots Q_n X_n \phi \tag{1}$$

where $n \geq 0$, $X_0, \dots, X_n$ are pairwise disjoint subsets of $\mathcal{A}$, every Q_i is either $\exists$ or $\forall$, and ϕ is a propositional formula over $\mathcal{A}$ in conjunctive normal form (CNF). QBFs as in (1) are in prenex conjunctive normal form. More general QBFs can be transformed to this form in a satisfiability-preserving way (Giunchiglia et al. 2009). Atoms in X_i are existentially (universally) quantified if Q_i is $\exists$ ($\forall$). Sequences of quantifiers and sets $Q_0 X_0 \dots Q_n X_n$ are called prefixes, and abbreviated by $\mathcal{Q}$. With it, a QBF as in (1) can be written as $\mathcal{Q}\phi$. As usual, we represent CNF formulas as sets of clauses, and clauses as sets of literals. For sets X and Y of atoms such that $X \subseteq Y$, we define $\mathit{fixbf}(X, Y)$ as the set of clauses $\{\{p\} \mid p \in X\} \cup \{\{\neg p\} \mid p \in Y \setminus X\}$ that selects models containing the atoms in X and no other atom from Y. That is, if ϕ is a formula then the models of $\phi \cup \mathit{fixbf}(X, Y)$ are $\{M \mid M$ is a model of ϕ and $M \cap Y = X\}$. Given that a CNF formula is satisfiable if it has some model, the satisfiability of a QBF can be defined as follows:

- $\exists X \phi$ is satisfiable if $\phi \cup \mathit{fixbf}(Y, X)$ is satisfiable for some $Y \subseteq X$.
- $\forall X \phi$ is satisfiable if $\phi \cup \mathit{fixbf}(Y, X)$ is satisfiable for all $Y \subseteq X$.
- $\exists X \mathcal{Q}\phi$ is satisfiable if $\mathcal{Q}\big(\phi \cup \mathit{fixbf}(Y, X)\big)$ is satisfiable for some $Y \subseteq X$.

- $\forall X \mathcal{Q}\phi$ is satisfiable if $\mathcal{Q}\big(\phi \cup \mathit{fixbf}(Y, X)\big)$ is satisfiable for all $Y \subseteq X$.

The formula ϕ in $\mathcal{Q}\phi$ generates a set of models, while the prefix $\mathcal{Q}$ can be interpreted as a kind of query over them. Consider $\phi_1 = \{\{a, b, \neg c\}, \{c\}\}$ and its models $\{\{a, c\}, \{a, b, c\}, \{b, c\}\}$. Adding the prefix $\mathcal{Q}_1 = \exists\{a\}\forall\{b\}$ amounts to querying if there is some subset Y_1 of $\{a\}$ such that for all subsets Y_2 of $\{b\}$ there is some model of ϕ_1 that contains the atoms in $Y_1 \cup Y_2$ and no other atoms from $\{a, b\}$. The answer is yes, for $Y_1 = \{a\}$, hence $\mathcal{Q}_1\phi_1$ is satisfiable. One can check that letting $\mathcal{Q}_2$ be $\exists\{b, c\}\forall\{a\}$ it holds that $\mathcal{Q}_2\phi_1$ is satisfiable, while letting $\mathcal{Q}_3$ be $\exists\{a\}\forall\{b, c\}$ we have that $\mathcal{Q}_3\phi_1$ is not.

3 Planning Problems

In this section, we define different planning problems with deterministic and non-concurrent actions using a transition function approach building on the work of Tu et al. (2007).

The domain of a planning problem is described in terms of fluents, i.e. properties changing over time, normal actions, and sensing actions for observing fluent values. We represent them by disjoint sets F, A_n, and A_s of atoms, respectively, let A be the set $A_n \cup A_s$ of actions, and assume that F and A are non-empty. For clarity, we denote sensing actions in A_s by a^f for some $f \in F$, indicating that a^f observes fluent f. To simplify the presentation, we assume that sets F, A_n, A_s and A are fixed. A state s is a set of fluents, $s \subseteq F$, that represents a snapshot of the domain. To describe planning domains, we have to specify what is the next state after the application of actions. Technically, this is done by a transition function Φ, that is, a function that takes as arguments a state and an action, and returns either one state or the bottom symbol $\bot$. Formally, $\Phi\colon \mathcal{P}(F) \times A \rightarrow \mathcal{P}(F) \cup \{\bot\}$, where $\mathcal{P}(F)$ denotes the power set of F. The case where $\Phi(s, a) = \bot$ represents that action a is not executable in state s.

Example. Let R be the set of rooms $\{1, \ldots, r\}$. We represent our example domain with the fluents $F = \{\mathit{at}(x), \mathit{clean}(x), \mathit{occupied}(x) \mid x \in R\}$, normal actions $A_n = \{\mathit{go}, \mathit{sweep}\}$, and sensing actions $A_s = \{\mathit{sense}(\mathit{occupied}(x)) \mid x \in R\}$. For $r = 2$, $s_1 = \{\mathit{at}(1), \mathit{clean}(1)\}$ is the state representing the initial situation of our classical planning example. The transition function Φ_e can be defined as follows: $\Phi_e(s, \mathit{go})$ is $(s \setminus \{\mathit{at}(x) \mid x \in R\}) \cup \{\mathit{at}(x+1) \mid \mathit{at}(x) \in s, x < r\} \cup \{\mathit{at}(r) \mid \mathit{at}(r) \in s\}$, $\Phi_e(s, \mathit{sweep})$ is $s \cup \{\mathit{clean}(x) \mid \mathit{at}(x) \in s\}$ if, for all $x \in R$, $\mathit{at}(x) \in s$ implies $\mathit{occupied}(x) \notin s$, and is $\bot$ otherwise; and, for all $x \in R$, $\Phi_e(s, \mathit{sense}(\mathit{occupied}(x)))$ is s if $\mathit{at}(x) \in s$ and is $\bot$ otherwise.

Once we have fixed the domain, we can define a planning problem as a tuple $\langle \Phi, I, G\rangle$ where Φ is a transition function, and $I, G \subseteq \mathcal{P}(F)$ are non-empty sets of initial and goal states. A conformant planning problem is a planning problem with no sensing actions, viz. $A_s = \emptyset$, and a classical planning problem is a conformant planning problem where I is a singleton. A planning problem with assumptions is then a tuple $\langle \Phi, I, G, \mathit{As}\rangle$ where $\langle \Phi, I, G\rangle$ is a planning problem and $\mathit{As} \subseteq F$ is a set of possible assumptions.

Example. The initial situation is $I_1 = \{s_1\}$ for our classical planning problem, $I_2 = \{\{\mathit{at}(1)\} \cup X \mid X \subseteq \{\mathit{clean}(1), \mathit{clean}(2)\}\}$ for the first conformant planning problem and $I_3 = \{X \cup Y \mid X \in I_2, Y \in \{\{\mathit{occupied}(1)\}, \{\mathit{occupied}(2)\}\}\}$ for the second one. All problems share the goal states $G_e = \{s \subseteq F \mid \text{ for all } x \in R \text{ either } \mathit{occupied}(x) \in s \text{ or } \mathit{clean}(x) \in s\}$. Let $\mathcal{PP}_1$ be $\langle \Phi_e, I_1, G_e\rangle$, $\mathcal{PP}_2$ be $\langle \Phi_e, I_2, G_e\rangle$, and $\mathcal{PP}_3$ be $\langle \Phi_e, I_3, G_e\rangle$. If we disregard sensing actions (and adapt Φ_e consequently) these problems correspond to our examples of classical and conformant planning, respectively. The one of assumption-based planning is given by $\mathcal{PP}_4 = \langle \Phi_e, I_3, G_e, \{occupied(1), occupied(2)\}\rangle$, and the one of conditional planning by $\mathcal{PP}_3$ with sensing actions.

Our next step is to define the solutions of a planning problem. For this, we specify what is a plan and extend transitions functions to apply to plans and sets of states. A plan and its length are defined inductively as follows:

- $[]$ is a plan, denoting the empty plan of length 0.
- If $a \in A_n$ is a non-sensing action and p is a plan, then $[a; p]$ is a plan of length one plus the length of p.
- If $a^f \in A_s$ is a sensing action, and p_f, $p_{\overline{f}}$ are plans, then $[a^f; (p_f, p_{\overline{f}})]$ is a plan of length one plus the maximum of the lengths of p_f and $p_{\overline{f}}$.

We simplify notation and write $[a; []]$ as $[a]$, and $[a; [\sigma]]$ as $[a; \sigma]$ for any action a and plan $[\sigma]$. For example, $p_1 = [go; sweep]$, $p_2 = [sweep; go; sweep]$, and $p_3 = [sense(occupied(1)); ([go; clean], [clean])]$ are plans of length 2, 3, and 3, respectively.

We extend the transition function Φ to a set of states S as follows: $\Phi(S, a)$ is $\bot$ if there is some $s \in S$ such that $\Phi(a, s) = \bot$, and is $\bigcup_{s\in S} \Phi(s, a)$ otherwise. In our example, $\Phi_e(I_1, go) = \{\{at(2), clean(1)\}\}$, $\Phi_e(I_2, sweep) = \{\{at(1), clean(1)\}, \{at(1), clean(1), clean(2)\}\}$, $\Phi_e(I_3, sweep) = \bot$, and $\Phi_e(\Phi_e(I_3, go), sweep) = \bot$. With this, we can extend the transition function Φ to plans as follows. Let p be a plan, and S a set of states, then:

- If $p = []$ then $\Phi(S, p) = S$.
- If $p = [a; q]$, where a is a non-sensing action and q is a plan, then

$$\Phi(S, p) = \begin{cases} \bot & \text{if } \Phi(S, a) = \bot \\ \Phi(\Phi(S, a), q) & \text{otherwise} \end{cases}$$

- If $p = [a^f; (q_f, q_{\overline{f}})]$, where a^f is a sensing action and q_f, $q_{\overline{f}}$ are plans, then

$$\Phi(S, p) = \begin{cases} \bot & \text{if either } \Phi(S, a^f), \Phi(S^f, q_f) \text{ or } \Phi(S^{\overline{f}}, q_{\overline{f}}) \text{ is } \bot \\ \Phi(S^f, q_f) \cup \Phi(S^{\overline{f}}, q_{\overline{f}}) & \text{otherwise} \end{cases}$$

  where $S^f = \{s \mid f \in s, s \in \Phi(S, a^f)\}$ and $S^{\overline{f}} = \{s \mid f \notin s, s \in \Phi(S, a^f)\}$.

In our example, $\Phi(I_1, p_1) = \{\{at(2), clean(1), clean(2)\}\}$, $\Phi(I_2, p_2) = \{\{at(2), clean(1), clean(2)\}\}$, $\Phi(I_3, q) = \bot$ for any plan q without sensing actions that involves some *sweep* action, $\Phi(\{s \in I_3 \mid occupied(1) \in s\}, p_1) = \{\{at(2), occupied(1), clean(2)\} \cup X \mid X \subseteq \{clean(1)\}\}$, and $\Phi(I_3, p_3) = \{\{at(2), occupied(1), clean(2)\} \cup X \mid X \subseteq \{clean(1)\}\} \cup \{\{at(1), occupied(2), clean(1)\} \cup X \mid X \subseteq \{clean(2)\}\}$.

We can now define the solutions of planning problems: a plan p is a solution to planning problem $\langle \Phi, I, G\rangle$ if $\Phi(I, p) \neq \bot$ and $\Phi(I, p) \subseteq G$. In our example, plan p_1 solves $\mathcal{PP}_1$, p_2 solves $\mathcal{PP}_2$, and p_3 solves $\mathcal{PP}_3$. There is no plan without sensing actions solving $\mathcal{PP}_3$. For assumption-based planning, a tuple $\langle p, T, F\rangle$, where p is a plan and $T, F \subseteq As$, is a solution to a planning problem with assumptions $\langle \Phi, I, G, As\rangle$ if (1) $J = \{s \mid s \in I, T \subseteq s, s \cap F = \emptyset\}$ is not empty, and (2) p solves the planning problem $\langle \Phi, J, G\rangle$. Condition (1) guarantees that the true assumptions T and the false assumptions F are consistent with some initial state, and condition (2) checks that p achieves the goal starting from all initial states consistent with the assumptions. For example, the planning problem with assumptions $\mathcal{PP}_4$ is solved by $\langle p_1, \{occupied(1)\}, \{\}\rangle$ and by $\langle [sweep], \{occupied(2)\}, \{\}\rangle$.

4 Representing Planning Problems in ASP

In this section we present an approach for representing planning problems using logic programs. Let F, A_n, A_s and A be sets of atoms as before, and let $F' = \{f' \mid f \in F\}$ be a set of atoms that we assume to be disjoint from the others. We use the atoms in F' to represent the value of the fluents in the previous situation. We represent planning problems by planning descriptions, that consist of dynamic rules to represent transition functions, initial rules to represent initial states, and goal rules to represent goal states. Formally, a dynamic rule is a rule whose head atoms belong to F and whose body atoms belong to $A \cup F \cup F'$, an initial rule is a rule whose atoms belong to F, and a goal rule is an integrity constraint whose atoms belong to F. Then a planning description $\mathcal{D}$ is a tuple $\langle DR, IR, GR\rangle$ of dynamic rules DR, initial rules IR, and goal rules GR. By $\mathbf{D}(\mathcal{D})$, $\mathbf{I}(\mathcal{D})$ and $\mathbf{G}(\mathcal{D})$ we refer to the elements DR, IR and GR, respectively.

Example. We represent the transition function Φ_e by the following dynamic rules DR_e:

$$
\begin{array}{rlrl}
at(R) & \leftarrow go, at(R-1)', R < r & clean(R) & \leftarrow sweep, at(R)' \\
at(r) & \leftarrow go, at(r)' & clean(R) & \leftarrow clean(R)' \\
at(R) & \leftarrow \neg go, at(R)' & occupied(R) & \leftarrow occupied(R)' \\
\bot & \leftarrow sweep, at(R)', occupied(R)' & \bot & \leftarrow sense(occupied(R)), \neg at(R)'
\end{array}
$$

On the left column, the normal rules describe the position of the robot depending on its previous position and the action go, while the integrity constraint below forbids the robot to sweep if it is in a room that is occupied. On the right column, the first two rules state when a room is clean, the third one expresses that rooms remain occupied if they were before, and the last integrity constraint forbids the robot to observe a room if it is not at it. For the initial states, I_1 is represented by the initial rules $IR_1 = \{at(1) \leftarrow; clean(1) \leftarrow\}$, I_2 is represented by IR_2:

$$
at(1) \leftarrow \qquad\qquad \{clean(R)\} \leftarrow R = 1..r
$$

and I_3 by IR_3, that contains the rules in IR_2 and also these ones:

$$
\{occupied(R)\} \leftarrow R = 1..r \qquad\qquad \bot \leftarrow \{occupied(R)\} \neq 1
$$

The choice rules generate the different possible initial states, and the integrity constraint inforces that exactly one room is occupied. The goal states G are represented by $GR_e = \{\bot \leftarrow \neg occupied(R), \neg clean(R), R = 1..n\}$ that forbids states where some room is not occupied and not clean.

Next we specify formally the relation between planning descriptions and planning problems. We extract a transition function $\Phi(s, a)$ from a set of dynamic rules DR by looking at the stable models of the program $s' \cup \{a \leftarrow\} \cup DR$ for every state s and action a, where s' is $\{f' \mid f \in s\}$. The state s is represented as s' to stand for the previous situation, and the action and the dynamic rules generate the next states. Given that we consider only deterministic transition functions, we restrict ourselves to deterministic problem descriptions, where we say that a set of dynamic rules DR is deterministic if for every state $s \subseteq F$ and action $a \in A$, the program $s' \cup \{a \leftarrow\} \cup DR$ has zero or one stable model, and a planning description $\mathcal{D}$ is deterministic if $\mathbf{D}(\mathcal{D})$ is deterministic. A deterministic set of dynamic rules DR defines the transition function

$$\Phi_{DR}(s,a) = \begin{cases} M \cap F & \text{if } s' \cup \{a\} \cup DR \text{ has a single stable model } M \\ \bot & \text{otherwise} \end{cases}$$

defined for every state $s \subseteq F$ and action $a \in A$. Note that given that DR is deterministic, the second condition only holds when the program $s' \cup \{a\} \cup DR$ has no stable models. For the case where no action occurs, we require dynamic rules to make the previous state persist. This condition is not strictly needed, but it makes the formulation of the solutions to planning problems in Section 6 easier. Formally, we say that a set of dynamic rules DR is inertial if for every state $s \subseteq F$ it holds that $SM(s' \cup DR) = \{s' \cup s\}$, and we say that a planning description $\mathcal{D}$ is inertial if $\mathbf{D}(\mathcal{D})$ is inertial. From now on, we restrict ourselves to deterministic and inertial planning descriptions.

Coming back to the initial and goal rules of a planning description $\mathcal{D}$, the first ones represent the initial states $SM(\mathbf{I}(\mathcal{D}))$, while the second ones represent the goal states $SM(\{\{f\} \leftarrow \mid f \in F\} \cup \mathbf{G}(\mathcal{D}))$. In the latter case, the choice rules generate all possible states while the integrity constraints in $\mathbf{G}(\mathcal{D})$ eliminate those that are not goal states. Given that we consider only non-empty subsets of initial and goal states, we require the programs $\mathbf{I}(\mathcal{D})$ and $\{\{f\} \leftarrow \mid f \in F\} \cup \mathbf{G}(\mathcal{D})$ to have at least one stable model. Finally, putting all together, we say that a deterministic and inertial planning description $\mathcal{D}$ represents the planning problem $\langle \Phi_{\mathbf{D}(\mathcal{D})}, SM(\mathbf{I}(\mathcal{D})), SM(\{\{f\} \leftarrow \mid f \in F\} \cup \mathbf{G}(\mathcal{D}))\rangle$. Moreover, the planning description $\mathcal{D}$ together with a set of atoms As represent the planning problem with assumptions $\langle \Phi_{\mathbf{D}(\mathcal{D})}, SM(\mathbf{I}(\mathcal{D})), SM(\{\{f\} \leftarrow \mid f \in F\} \cup \mathbf{G}(\mathcal{D})), As\rangle$.

Example. One can check that the dynamic rules DR_e are deterministic, inertial, and define the transition function Φ_e of the example. Moreover, $\mathcal{D}_1 = \langle DR_e, IR_1, GR_e\rangle$ represents $\mathcal{PP}_1$, $\mathcal{D}_2 = \langle DR_e, IR_2, GR_e\rangle$ represents $\mathcal{PP}_2$, $\mathcal{D}_3 = \langle DR_e, IR_3, GR_e\rangle$ represents $\mathcal{PP}_3$, and $\mathcal{D}_3$ with the set of atoms $\{\mathit{occupied}(1), \mathit{occupied}(2)\}$ represents $\mathcal{PP}_4$.

5 Quantified Answer Set Programming

Quantified Answer Set Programming (QASP) is an extension of ASP to quantified logic programs (QLPs), analogous to the extension of propositional formulas by QBFs. A quantified logic program over $\mathcal{A}$ has the form

$$Q_0 X_0 \dots Q_n X_n P \tag{2}$$

where $n \geq 0$, $X_0, \dots, X_n$ are pairwise disjoint subsets of $\mathcal{A}$, every Q_i is either $\exists$ or $\forall$, and P is a logic program over $\mathcal{A}$. Prefixes are the same as in QBFs, and we may refer to a QLP as in (2) by $\mathcal{Q}P$. For sets X and Y of atoms such that $X \subseteq Y$, we define $\mathit{fixcons}(X,Y)$ as the set of rules $\{\bot \leftarrow \neg x \mid x \in X\} \cup \{\bot \leftarrow x \mid x \in Y \setminus X\}$ that selects stable models containing the atoms in X and no other atom from Y. That is, if P is a logic program then the stable models of $P \cup \mathit{fixcons}(X,Y)$ are $\{M \mid M \text{ is a stable model of } P \text{ and } M \cap Y = X\}$. Given that a logic program is satisfiable if it has a stable model, the satisfiability of a QLP is defined as follows:

- $\exists X P$ is satisfiable if program $P \cup \mathit{fixcons}(Y,X)$ is satisfiable for some $Y \subseteq X$.
- $\forall X P$ is satisfiable if program $P \cup \mathit{fixcons}(Y,X)$ is satisfiable for all $Y \subseteq X$.

- $\exists X \mathcal{Q} P$ is satisfiable if program $\mathcal{Q}\big(P \cup \mathit{fixcons}(Y, X)\big)$ is satisfiable for some $Y \subseteq X$.
- $\forall X \mathcal{Q} P$ is satisfiable if program $\mathcal{Q}\big(P \cup \mathit{fixcons}(Y, X)\big)$ is satisfiable for all $Y \subseteq X$.

As with QBFs, program P in $\mathcal{Q}P$ generates a set of stable models, while its prefix $\mathcal{Q}$ can be seen as a kind of query on it. Consider $P_1 = \{\{a\} \leftarrow ; \{b\} \leftarrow ; c \leftarrow a; c \leftarrow b; \bot \leftarrow \neg c\}$ and its stable models $\{a, c\}$, $\{a, b, c\}$, $\{b, c\}$. The prefixes of $\mathcal{Q}_1 P_1$, $\mathcal{Q}_2 P_1$ and $\mathcal{Q}_3 P_1$ pose the same queries over the stable models of P than those posed in $\mathcal{Q}_1 \phi_1$, $\mathcal{Q}_2 \phi_1$ and $\mathcal{Q}_3 \phi_1$ over the models of ϕ_1. Given that the stable models of P_1 and the models of ϕ_1 coincide, the satisfiability of the $\mathcal{Q}_i P_1$'s is the same as that of the corresponding $\mathcal{Q}_i \phi_1$'s. This result is generalized by the following theorem that relates QLPs and QBFs.

Theorem 5.1
Let P be a logic program over $\mathcal{A}$ and ϕ be a CNF formula over $\mathcal{A} \cup \mathcal{B}$ such that $SM(P) = \{M \cap \mathcal{A} \mid M \text{ is a model of } \phi\}$. For every prefix $\mathcal{Q}$ whose sets belong to $\mathcal{A}$, the QLP $\mathcal{Q}P$ is satisfiable if and only if the QBF $\mathcal{Q}\phi$ is satisfiable.

The proof is by induction on the number of quantifiers in $\mathcal{Q}$. The condition $SM(P) = \{M \cap \mathcal{A} \mid M \text{ is a model of } \phi\}$ of Theorem 5.1 is satisfied by existing polynomial-time translations from logic programs P over $\mathcal{A}$ to CNF formulas ϕ over $\mathcal{A} \cup \mathcal{B}$, and from CNF formulas ϕ over $\mathcal{A}$ to logic programs P over $\mathcal{A}$ (Janhunen 2004). Using these translations, Theorem 5.1, and the fact that deciding whether a QBF is satisfiable is PSPACE-complete, we can prove the following complexity result about QASP.

Theorem 5.2
The problem of deciding whether a given QLP $\mathcal{Q}P$ is satisfiable is PSPACE-complete.

The implementation of our system QASP2QBF[1] relies on the previous results. The input is a QLP $\mathcal{Q}P$ that is specified by putting together the rules of P with facts over the predicates $\_exists/2$ and $\_forall/2$ describing the prefix $\mathcal{Q}$, where $\_exists(i, a)$ ($\_forall(i, a)$, respectively) asserts that the atom a is existentially (universally, respectively) quantified at position i of $\mathcal{Q}$. The system first translates P into a CNF formula ϕ that satisfies the condition of Theorem 5.1 using the tools LP2NORMAL, LP2ACYC, and LP2SAT,[2] and then uses a QBF solver to decide the satisfiability of $\mathcal{Q}\phi$. If $\mathcal{Q}\phi$ is satisfiable and the outermost quantifier is existential, then the system returns an assignment to the atoms occurring in the scope of that quantifier.

6 Solving Planning Problems in QASP

In this section, we describe how to solve planning problems represented by a planning description using QASP.

Classical planning. We start with a planning description $\mathcal{D}$ that represents a classical planning problem $\mathcal{PP} = \langle \Phi, \{s_0\}, G \rangle$. Our task, given a positive integer n, is to find a plan $[a_1; \ldots; a_n]$ of length n such that $\Phi(\{s_0\}, p) \subseteq G$. This can be solved as usual in answer set planning (Lifschitz 2002), using choice rules to generate possible plans, initial rules to define the initial state of the problem, dynamic rules replicated n times to define the next n steps, and goal rules to check the goal conditions at the last step. To do that

[1] `https://github.com/potassco/qasp2qbf`
[2] `http://research.ics.aalto.fi/software/asp`

in this context, we first let *Domain* be the union of the following sets of facts asserting the time steps of the problem, the actions, and the fluents that are sensed by each sensing action: $\{t(T) \mid t \in \{1,\ldots,n\}\}$, $\{action(a) \mid a \in A\}$, and $\{senses(a^f, f) \mid a^f \in A_s\}$. The last set is only needed for conditional planning, but we already add it here for simplicity. Given these facts, the following choice rule generates the possible plans of the problem:

$$\{occ(A,T) : action(A)\} = 1 \leftarrow t(T) \tag{3}$$

Additionally, by $\mathcal{D}^\circ_{\mathbf{I}}$ ($\mathcal{D}^\circ_{\mathbf{G}}$, respectively) we denote the set of rules that results from replacing in $\mathbf{I}(\mathcal{D})$ ($\mathbf{G}(\mathcal{D})$, respectively) the atoms $f \in F$ by $h(f,0)$ (by $h(f,n)$, respectively); by $\mathcal{D}^\circ_{\mathbf{D}}$ we denote the set of rules that results from replacing in $\mathbf{D}(\mathcal{D})$ the atoms $f \in F$ by $h(f,T)$, the atoms $f' \in F'$ by $h(f,T-1)$, the atoms $a \in A$ by $occ(a,T)$ and adding the atom $t(T)$ to the body of every rule; and by $\mathcal{D}^\circ$ we denote the program $\mathcal{D}^\circ_{\mathbf{I}} \cup \mathcal{D}^\circ_{\mathbf{D}} \cup \mathcal{D}^\circ_{\mathbf{G}}$. Putting all together, the program $Domain \cup (3) \cup \mathcal{D}^\circ$ represents the solutions to the planning problem $\mathcal{PP}$. The choice rule (3) guesses plans $[a_1;\ldots;a_n]$ using atoms of the form $occ(a_1,1),\ldots,occ(a_n,n)$, the rules of $\mathcal{D}^\circ_{\mathbf{I}}$ define the initial state s_0 using atoms of the form $h(\cdot,0)$, the rules of $\mathcal{D}^\circ_{\mathbf{D}}$ define the next states $s_i = \Phi(s_{i-1},a_i)$ for $i \in \{1,\ldots,n\}$ using atoms of the form $h(\cdot,i)$ while at the same time check that the actions a_i are executable in s_{i-1}, and the rules of $\mathcal{D}^\circ_{\mathbf{G}}$ check that the last state s_n belongs to G. Of course, all this works only because $\mathcal{D}$ represents $\mathcal{PP}$ and therefore Φ, $\{s_0\}$ and G are defined by $\mathbf{D}(\mathcal{D})$, $\mathbf{I}(\mathcal{D})$ and $\mathbf{G}(\mathcal{D})$, respectively. Finally, letting *Occ* be the set of atoms $\{occ(a,t) \mid a \in A, t \in \{1,\ldots,n\}\}$, we can represent the solutions to $\mathcal{PP}$ by the quantified logic program

$$\exists Occ\big(Domain \cup (3) \cup \mathcal{D}^\circ\big) \tag{4}$$

where the atoms selected by the existential quantifier correspond to solutions to $\mathcal{PP}$. Going back to our example, where $\mathcal{D}_1$ represents the problem $\mathcal{PP}_1$, we have that for $n=2$ the program (4) (adapted to $\mathcal{D}_1$) is satisfiable selecting the atoms $\{occ(go,1), occ(sweep,2)\}$ that represent the solution p_1.

Conformant planning. When $\mathcal{D}$ represents a conformant planning problem $\mathcal{PP} = \langle \Phi, I, G\rangle$ our task is to find a plan p such that $\Phi(I,p) \subseteq G$, or, alternatively, p must be such that for all $s \in I$ it holds that $\Phi(\{s\},p) \subseteq G$. This formulation of the problem suggests to use a prefix $\exists\forall$ where the existential quantifier guesses a plan p and the universal quantifier considers all initial states. Let us make this more concrete. From now on we assume that $\mathbf{I}(\mathcal{D})$ is in *GDT* form.[3] If that is not the case then we can translate the program into *GDT* form using the method mentioned in the Background section, and we continue from there. Let *Gen*, *Def* and *Test* be the choice rules, normal rules and integrity constraints of $\mathcal{D}^\circ_{\mathbf{I}}$, respectively, and let *Open* be the set $\{h(f,0) \mid \{h(f,0)\} \leftarrow \in Gen\}$ of possible guesses of the choice rules of $\mathcal{D}^\circ_{\mathbf{I}}$. Observe that for every possible set $X \subseteq Open$ the program $X \cup Def$ has a unique stable model M, and if M is also a model of *Test* then M is a stable model of $\mathcal{D}^\circ_{\mathbf{I}}$. Moreover, note that all stable models of $\mathcal{D}^\circ_{\mathbf{I}}$ can be constructed in this manner. Given this, we say that the sets $X \subseteq Open$ that lead to a stable model M of $\mathcal{D}^\circ_{\mathbf{I}}$ are relevant, because they can be used as representatives of the initial states, and the other sets in *Open* are irrelevant. Back to our quantified logic program, we are going to use the prefix $\exists Occ \forall Open$. This works well with the logic program $Domain \cup (3) \cup \mathcal{D}^\circ$

[3] Note that this implies that $\mathcal{D}^\circ_{\mathbf{I}}$ is also in *GDT* form.

whenever all choices $X \subseteq \mathit{Open}$ are relevant. But it fails as soon as there are irrelevant sets because, when we select them as subsets of *Open* in the universal quantifier, the resulting program becomes unsatisfiable. To fix this, we can modify our logic program so that for the irrelevant sets the resulting program becomes always satisfiable. We do that in two steps. First, we modify $\mathcal{D}^{\circ}_{\mathbf{I}}$ so that the irrelevant sets lead to a unique stable model that contains the special atom $\alpha(0)$. This is done by the program $\mathcal{D}^{\bullet}_{\mathbf{I}}$ that results from replacing in $\mathcal{D}^{\circ}_{\mathbf{I}}$ the symbol $\bot$ in the head of the integrity constraints by $\alpha(0)$. Additionally, we consider the rule

$$\alpha(T) \leftarrow t(T), \alpha(T{-}1) \tag{5}$$

that derives $\alpha(1), \ldots, \alpha(n)$ for the irrelevant sets. Second, we modify $\mathcal{D}^{\circ}_{\mathbf{D}}$ and $\mathcal{D}^{\circ}_{\mathbf{G}}$ so that whenever those special atoms are derived, these programs are inmediately satisfied. This is done by the programs $\mathcal{D}^{\bullet}_{\mathbf{D}}$ and $\mathcal{D}^{\bullet}_{\mathbf{G}}$ that result from adding the literal $\neg\alpha(T)$ to the bodies of the rules in $\mathcal{D}^{\circ}_{\mathbf{D}}$ and $\mathcal{D}^{\circ}_{\mathbf{G}}$, respectively. Whenever the special atoms $\alpha(0)$, $\ldots$, $\alpha(n)$ are derived, they deactivate the rules in $\mathcal{D}^{\bullet}_{\mathbf{D}}$ and $\mathcal{D}^{\bullet}_{\mathbf{G}}$ and make the program satisfiable. We denote by $\mathcal{D}^{\bullet}$ the program $\mathcal{D}^{\bullet}_{\mathbf{I}} \cup \mathcal{D}^{\bullet}_{\mathbf{D}} \cup \mathcal{D}^{\bullet}_{\mathbf{G}}$. Then the following theorem establishes the correctness and completeness of the approach.

Theorem 6.1

Let $\mathcal{D}$ be a planning description that represents a conformant planning problem $\mathcal{PP}$, and n be a positive integer. If $\mathbf{I}(P)$ is in *GDT* form, then there is a plan of length n that solves $\mathcal{PP}$ if and only if the following quantified logic program is satisfiable:

$$\exists \mathit{Occ} \forall \mathit{Open}\big(\mathit{Domain} \cup \mathcal{D}^{\bullet} \cup (3) \cup (5)\big) \tag{6}$$

In our example, where $\mathcal{D}_2$ represents the problem $\mathcal{PP}_2$, for $n = 3$ the program (6) (adapted to $\mathcal{D}_2$) is satisfiable selecting the atoms $\{\mathit{occ}(\mathit{sweep}, 1), \mathit{occ}(\mathit{go}, 2), \mathit{occ}(\mathit{sweep}, 3)\}$ that represent the solution p_2, while for $\mathcal{D}_3$, that represents $\mathcal{PP}_3$, the corresponding program is unsatisfiable for any integer n.

Assumption-based planning. Let $\mathcal{D}$, together with a set of atoms $\mathit{As} \subseteq F$, represent a conformant planning problem with assumptions $\mathcal{PP} = \langle \Phi, I, G, \mathit{As}\rangle$. To solve this problem we have to find a plan $p = [a_1; \ldots; a_n]$ and a set of assumptions $T, F \subseteq \mathit{As}$ such that (1) the set $J = \{s \mid s \in I, T \subseteq s, s \cap F = \emptyset\}$ is not empty, and (2) p solves the conformant planning problem $\langle \Phi, J, G\rangle$. The formulation of the problem suggests that first we can guess the set of assumptions T and F, and then check (1) and (2) separately. The guess can be represented by the set of rules *Guess* that consists of the facts $\{\mathit{assumable}(f) \mid f \in \mathit{As}\}$ and the choice rule

$$\{\mathit{assume}(F, \mathit{true}); \mathit{assume}(F, \mathit{false})\} \leq 1 \leftarrow \mathit{assumable}(F)$$

that generates all possible sets of assumptions using the predicate $\mathit{assume}/2$. Moreover, we add the set of atoms $\mathit{Assume} = \{\mathit{assume}(f, v) \mid f \in \mathit{As}, v \in \{\mathit{true}, \mathit{false}\}\}$ to *Occ* at the outermost existential quantifier of our program. Condition (1) can be checked by the set of rules *C1* that can be divided in two parts. The first part is a copy of the initial rules, that consists of the rules that result from replacing in $\mathbf{I}(P)$ every atom $f \in F$ by $\mathit{init}(f)$, and in this way generates all possible initial states in I using the predicate $\mathit{init}/1$. The second part contains the integrity constraints

$$\bot \;\leftarrow\; \neg \mathit{init}(F), \mathit{assume}(F, \mathit{true}) \qquad\qquad \bot \;\leftarrow\; \mathit{init}(F), \mathit{assume}(F, \mathit{false})$$

that guarantee that the guessed assumptions represented by $assume/2$ hold in some initial state represented by $init/1$. Condition (2) can be represented extending the program for conformant planning with the following additional rules *C2*, stating that the initial states that do not agree with the guessed assumptions are irrelevant:

$$\alpha(0) \quad \leftarrow \quad \neg h(F,0), assume(F,true) \qquad\qquad \alpha(0) \quad \leftarrow \quad h(F,0), assume(F,false)$$

With these rules, the plans only have to succeed starting from the initial states that agree with the assumptions, and condition (2) is satisfied.

Theorem 6.2

Let $\mathcal{D}$, $\mathcal{PP}$, and As be as specified before, and n be a positive integer. If $\mathbf{I}(\mathcal{D})$ is in *GDT* form, then there is a plan with assumptions of length n that solves $\mathcal{PP}$ if and only if the following quantified logic program is satisfiable:

$$\exists(Occ \cup Assume)\forall Open\big(Domain \cup \mathcal{D}^{\bullet} \cup (3) \cup (5) \cup Guess \cup C1 \cup C2\big) \qquad (7)$$

In our example, where $\mathcal{D}_3$ together with the set of atoms $\{occupied(1), occupied(2)\}$ represents $\mathcal{PP}_4$, we have that for $n = 2$ the program (7) (adapted to $\mathcal{D}_3$) is satisfiable selecting the atoms $\{occ(go,1), occ(sweep,2), assume(occupied(1), true)\}$ that represent the solution $\langle p_1, \{occupied(1)\}, \{\}\rangle$, and for $n = 1$ selecting the atoms $\{occ(sweep,1), assume(occupied(2), true)\}$ that represent the solution $\langle [sweep], \{occupied(2)\}, \{\}\rangle$.

Conditional planning. Consider the case where $\mathcal{D}$ represents a planning problem with sensing actions $PP = \langle \Phi, I, G\rangle$. Again, we have to find a plan p such that $\Phi(I,p) \subseteq G$, but this time p has the form of a tree where sensing actions can be followed by different actions depending on sensing results. This suggests a formulation where we represent the sensing result at time point T by the truth value of an atom $obs(true,T)$, we guess those possible observations with a choice rule:

$$\{obs(true,T)\} \leftarrow t(T), T < n \qquad (8)$$

and, letting Obs_t be $\{obs(true,t)\}$ and Occ_t be $\{occ(a,t) \mid a \in A\}$ for $t \in \{1,\ldots,n\}$, we consider a QLP of the form

$$\exists Occ_1 \forall Obs_1 \ldots \exists Occ_{n-1} \forall Obs_{n-1} \exists Occ_n \forall Open\big(Domain \cup \mathcal{D}^{\bullet} \cup (3) \cup (5) \cup (8)\big).$$

This formulation nicely allows for a different plan $[a_1; \ldots; a_n]$ for every sequence $[O_1; \ldots; O_{n-1}]$ of observations $O_i \subseteq Obs_i$. But at the same time it requires that each such plan achieves the goal for all possible initial situations, and this requirement is too strong. Actually, we only want the plans to work for those cases where, at every time step T, the result of a sensing action a^f, represented by $obs(true,T)$, coincides with the value of the sensed fluent represented by $h(f,T)$. We can achieve this by signaling the other cases as irrelevant with the following rule:

$$\alpha(T) \leftarrow t(T), occ(A,T{-}1), senses(A,F), \{h(F,T{-}1), obs(true,T{-}1)\} = 1 \qquad (9)$$

where the cardinality constraint $\{h(F,T{-}1), obs(true,T{-}1)\} = 1$ holds if the truth value of $obs(true,T{-}1)$ and $h(F,T{-}1)$ is not the same. Another issue with the previous QLP is that it allows normal actions at every time point T to be followed by different actions at $T{+}1$ for each value of $obs(true,T)$. This is not a problem for the correctness of the approach, but it is not a natural representation. To fix this, we can consider that, whenever a normal action occurs at time point T, the case where $obs(true, T)$ holds is irrelevant:

$$\alpha(T) \leftarrow t(T), occ(A,T), \{senses(A,F)\} = 0, obs(true,T) \tag{10}$$

Appart from this, note that in conditional planning the different subplans may have different lengths. For this reason, in the choice rule (3) we have to replace the symbol '=' by '$\leq$'. We denote the new rule by $(3)^{\leq}$.

Theorem 6.3
Let $\mathcal{D}$ and $\mathcal{PP}$ be as specified before, and n be a positive integer. If $\mathbf{I}(\mathcal{D})$ is in *GDT* form, then there is a plan of length less or equal than n that solves $\mathcal{PP}$ if and only if the following quantified logic program is satisfiable:

$$\exists Occ_1 \forall Obs_1 \ldots \exists Occ_{n-1} \forall Obs_{n-1} \exists Occ_n \forall Open\big(Domain \cup P^{\bullet} \cup (3)^{\leq} \cup (5) \cup (8-10)\big) \tag{11}$$

For $\mathcal{D}_3$, that represents the problem $\mathcal{PP}_3$, and $n = 3$, the program (11) (adapted to $\mathcal{D}_3$) is satisfiable selecting first $\{occ(sense(occupied(1)))\}$ at Occ_1, then at Occ_2 selecting $\{occ(clean, 2)\}$ for the subset $\{\} \subseteq Obs_1$ and $\{occ(go, 2)\}$ for the subset $Obs_1 \subseteq Obs_1$, and finally at Occ_3 selecting $\{\}$ in all cases except for the subsets $Obs_1 \subseteq Obs_1$ and $\{\} \subseteq Obs_2$ that we select $\{occ(clean, 3)\}$. This assignment corresponds to plan p_3.

7 Experiments

We evaluate the performance of QASP2QBF in conformant and conditional planning benchmarks. We consider the problem OPT of computing a plan of optimal length. To solve it we first run the solver for length 1, and successively increment the length by 1 until an optimal plan is found. The solving times of this procedure are usually dominated by the unsatisfiable runs. To complement this, we also consider the problem SAT of computing a plan of a fixed given length, for which we know that a solution exists.

We evaluate QASP2QBF 1.0 and combine it with 4 differents QBF solvers:[4] CAQE 4.0.1 (Rabe and Tentrup 2015), DEPQBF 6.03 (Lonsing and Egly 2017), QESTO 1.0 (Janota and Marques-Silva 2015) and QUTE 1.1 (Peitl et al. 2019); and either none or one of 3 preprocessors: BLOQQER $v37$ (B; Biere et al. 2011), HQSPRE 1.4 (H; Wimmer et al. 2017), and QRATPRE+ 2.0 (Q; Lonsing and Egly 2019); for a total of 16 configurations. By CAQE$^{\circ}$ we refer to the combination of QASP2QBF with CAQE without preprocessor, and by CAQE^{B} to QASP2QBF with CAQE and the preprocessor BLOQQER. We proceed similarly for other QBF solvers and preprocessors. We compare QASP2QBF with the incomplete planner CPASP (Tu et al. 2007), that translates a planning description into a normal logic program that is fed to an ASP solver. In the experiments we have used the ASP solver CLINGO (version 5.5)[5] and evaluated its 6 basic configurations (CRAFTY (C), FRUMPY (F), HANDY (H), JUMPY (J), TRENDY (R), and TWEETY (T)). We refer to CPASP with CLINGO and configuration CRAFTY by CLINGO^{C}, and similarly with the others.

We use the benchmark set from Tu et al. (2007), but we have increased the size of the instances of some domains if they were too easy. The conformant domains are six variants of Bomb in the Toilet (Bt, Bmt, Btc, Bmtc, Btuc, Bmtuc), Domino and Ring. We have

[4] We follow the selection of QBF solvers and preprocessors of Mayer-Eichberger and Saffidine (2020).
[5] `https://potassco.org/clingo`

also added a small variation of the Ring domain, called Ringu, where the room of the agent is unknown, and the planner CPASP cannot find any plan due to its incompleteness. The conditional domains are four variations of Bomb in the Toilet with Sensing Actions (Bts1, Bts2, Bts3 and Bts4), Domino, Medical Problem (Med), Ring and Sick. All domains have 5 instances of increasing difficulty, except Domino in conformant planning that has 6, and Ring in conditional planning that has 4. For the problem SAT, the fixed plan length is always the minimal plan length for CPASP.

All experiments ran on an Intel Xeon 2.20GHz processor under Linux. Each run was limited to 30 minutes runtime and 16 GB of memory. We report the aggregated results per domain: average runtime in seconds and number of timeouts in parentheses for OPT, next to the average runtime in seconds for SAT, for which there were very few timeouts. To calculate the averages, we add 1800 seconds for every timeout. In Appendix A, we report these results for every solver and configuration, and provide further details. Here, we show and discuss the best configuration for each solver, separately for conformant planning in Table 1, and for conditional planning in Table 2.

In conformant planning, looking at the QASP2QBF configurations, for the variations of Bt, CAQE^{Q} and QESTO^{Q} perform better than DEPQBF^{Q} and QUTE^{H}. Domino is solved very quickly by all solvers, while in Ring and Ringu CAQE^{Q} clearly outperforms the others. The planner CLINGO^{J}, in the variations of Bt and Domino, in OPT has a similar performance to the best QASP2QBF solvers, while in SAT it is much faster and solves the problems in less than a second. In Ring, however, its performance is worse than that of CAQE^{Q}. Finally, in Ringu for OPT, given the incompleteness of the system, it never manages to find a plan and always times out. In conditional planning, CAQE^{B} is the best for the variations of Bts, while CLINGO^{H} is better than the other solvers for SAT but worse for OPT. In Domino, for OPT, the QASP2QBF solvers perform better than CLINGO^{H}, while for SAT only QESTO^{B} matches its performance. Finally, for Med, Ring, and Sick, all solvers yield similar times. Summing up, we can conclude that QASP2QBF with the right QBF solver and preprocessor compares well to CPASP, except for the SAT problem in conformant planning, while on the other hand it can solve problems, like Ringu, that are out of reach for CPASP due to its incompleteness.

	CLINGO^{J}		CAQE^{Q}		DEPQBF^{Q}		QESTO^{Q}		QUTE^{H}	
Bt	43 (0)	0	46 (0)	0	319 (0)	0	57 (0)	0	363 (1)	42
Bmt	50 (0)	0	155 (0)	2	367 (1)	2	76 (0)	1	382 (1)	12
Btc	258 (0)	0	413 (1)	1	566 (1)	2	394 (1)	1	729 (2)	38
Bmtc	744 (2)	0	771 (2)	9	1083 (3)	9	826 (2)	9	1082 (3)	374
Btuc	245 (0)	0	400 (1)	1	538 (1)	1	391 (1)	2	731 (2)	38
Bmtuc	739 (2)	0	780 (2)	14	1085 (3)	12	813 (2)	12	1440 (4)	378
Domino	0 (0)	0	0 (0)	0	0 (0)	0	0 (0)	0	0 (0)	0
Ring	673 (1)	416	281 (0)	40	1120 (3)	1109	790 (2)	392	1441 (4)	1440
Ringu	1800 (5)	0	246 (0)	50	1800 (5)	1800	1800 (5)	1800	1800 (5)	1800
Total	505 (10)	46	343 (6)	12	764 (17)	326	571 (13)	246	885 (22)	458

Table 1: Experimental results on conformant planning

	CLINGO[H]		CAQE[B]		DEPQBF[B]		QESTO[B]		QUTE[B]	
Bts1	795 (2)	49	14 (0)	3	364 (1)	361	102 (0)	65	365 (1)	361
Bts2	787 (2)	26	14 (0)	3	369 (1)	41	370 (1)	182	376 (1)	368
Bts3	802 (2)	82	15 (0)	4	372 (1)	10	371 (1)	364	380 (1)	370
Bts4	833 (2)	76	17 (0)	4	368 (1)	362	379 (1)	89	390 (1)	377
Domino	906 (2)	6	362 (1)	125	361 (1)	48	363 (1)	5	361 (1)	360
Med	0 (0)	0	3 (0)	1	3 (0)	1	3 (0)	1	3 (0)	1
Ring	902 (2)	900	902 (2)	640	901 (2)	900	902 (2)	900	901 (2)	900
Sick	0 (0)	0	1 (0)	0	1 (0)	0	1 (0)	1	1 (0)	0
Total	628 (12)	142	166 (3)	97	342 (7)	215	311 (6)	200	347 (7)	342

Table 2: Experimental results on conditional planning

8 Related Work

Conformant and conditional planning have already been addressed with ASP (Eiter et al. 2003; Son et al. 2005; Tu et al. 2007; Tu et al. 2011). Eiter et al. (2003) introduce the system $\texttt{dlv}^{\mathcal{K}}$ for planning with incomplete information. It solves a conformant planning problem by first generating a potential plan and then verifying it, no sensing actions are considered. Son et al. (2005; 2007; 2011) propose an approximation semantics for reasoning about action and change in the presence of incomplete information and sensing actions. This is then used for developing ASP-based conformant and conditional planners, like CPASP, that are generally incomplete.

Closely related is SAT-based conformant planning: C-PLAN (Castellini et al. 2003) is similar to $\texttt{dlv}^{\mathcal{K}}$ in identifying a potential plan before validating it. COMPILE-PROJECT-SAT (Palacios and Geffner 2005) uses a single call to a SAT-solver to compute a conformant plan. Their validity check is doable in linear time, if the planning problem is encoded in deterministic decomposable negation normal form. Unlike this, QBFPLAN (Rintanen 1999) maps the problem into QBF and uses a QBF-solver as back-end.

A more recent use of ASP for computing conditional plans is proposed by Yalciner et al. (2017). The planner deals with sensing actions and incomplete information; it generates multiple sequential plans before combining them in a graph representing a conditional plan. Cardinality constraints, defaults, and choices are used to represent the execution of sensing actions, their effects, and branches in the final conditional plan. In addition, the system computes sequential plans in parallel and also avoids regenerating plans.

Assumption-based planning, as considered here, is due to Davis-Mendelow et al. (2013). In that work, the problem is solved by translating it into classical planning using an adaptation of the translation of Palacios and Geffner (2009). To the best of our knowledge, there exists no ASP-based planner for this type of problems.

There are a number of extensions of ASP to represent problems whose complexity lays beyond NP in the polynomial hierarchy. A comprehensive review was made by Amendola et al. (2019), that presents the approach that is closer to QASP, named ASP with Quantifiers (ASP(Q)). Like QASP, it introduces existential and universal quantifiers, but they range over stable models of logic programs and not over atoms. This quantification over stable models is very useful for knowledge representation. For example, it allows us to represent conformant planning problems without the need of additional α atoms, using the following ASP(Q) program:

$$\exists^{st}\big(\mathit{Domain} \cup (3)\big)\ \forall^{st}\mathcal{D}^{\circ}_{\mathbf{I}}\ \exists^{st}\big(\mathcal{D}^{\circ}_{\mathbf{D}} \cup \mathcal{D}^{\circ}_{\mathbf{G}}\big) \tag{12}$$

where $\exists^{st}$ and $\forall^{st}$ are existential and universal stable model quantifiers, respectively. The program is coherent (or satisfiable, in our terms) if there is some stable model M_1 of $Domain \cup (3)$ such that for all stable models M_2 of $M_1 \cup \mathcal{D}^{\circ}_{\mathbf{I}}$ there is some stable model of $M_2 \cup \mathcal{D}^{\circ}_{\mathbf{D}} \cup \mathcal{D}^{\circ}_{\mathbf{G}}$. Stable models M_1 correspond to possible plans. They are extended in M_2 by atoms representing initial states, that are used in $M_2 \cup \mathcal{D}^{\circ}_{\mathbf{D}} \cup \mathcal{D}^{\circ}_{\mathbf{G}}$ to check if the plans achieve the goal starting from *all* those initial states. Assumption-based planning can be represented in a similar way, while for conditional planning we have not been able to come up with any formulation that does not use additional α atoms.

As part of this work, we have developed translations between QASP and ASP(Q). We leave their formal specification to Appendix B, and illustrate them here for conformant planning. From ASP(Q) to QASP, we assume that $\mathcal{D}^{\circ}_{\mathbf{I}}$ is in GDT form, and otherwise we translate it into this form. Then, the translation of an ASP(Q) program of the form (12) yields a QLP that is essentially the same as (6), except for some renaming of the additional α atoms and some irrelevant changes in the prefix. In the other direction, the QLP program (6) is translated to the ASP(Q) program

$$\exists^{st} P_0 \forall^{st} P_1 \exists^{st} \big(Domain \cup \mathcal{D}^{\bullet} \cup (3) \cup (5) \cup O \big) \tag{13}$$

where P_0 is $\{\{p'\} \leftarrow \mid p \in Occ\}$, P_1 is $\{\{p'\} \leftarrow \mid p \in Open\}$, and O contains the set of rules $\bot \leftarrow p, \neg p'$ and $\bot \leftarrow \neg p, p'$ for every $p \in Occ \cup Open$. Programs P_0 and P_1 guess the values of the atoms of the prefix using additional atoms p', and the constraints in O match those guesses to the corresponding original atoms.

9 Conclusion

We defined a general ASP language to represent a wide range of planning *problems*: classical, conformant, with assumptions, and conditional with sensing actions. We then defined a quantified extension of ASP, viz. QASP, to represent the *solutions* to those planning problems. Finally, we implemented and evaluated a QASP solver, available at `potassco.org`, to *compute* the solutions to those planning problems. Our focus lays on the generality of the language and the tackled problems; on the formal foundations of the approach, by relating it to simple transition functions; and on having a baseline implementation, whose performance we expect to improve further in the future.

References

Amendola, G., Ricca, F., and Truszczynski, M. 2019. Beyond NP: quantifying over answer sets. *Theory and Practice of Logic Programming 19,* 5-6, 705–721.

Biere, A., Lonsing, F., and Seidl, M. 2011. Blocked clause elimination for QBF. In *Proceedings of CADE'11*, Lecture Notes in Computer Science, vol. 6803. Springer-Verlag, 101–115.

Cabalar, P., Kaminski, R., Schaub, T., and Schuhmann, A. 2018. Temporal answer set programming on finite traces. *Theory and Practice of Logic Programming 18,* 3-4, 406–420.

Castellini, C., Giunchiglia, E., and Tacchella, A. 2003. SAT-based planning in complex domains: Concurrency, constraints and nondeterminism. *Artificial Intelligence 147,* 1-2, 85–117.

Davis-Mendelow, S., Baier, J., and McIlraith, S. 2013. Assumption-based planning: Generating plans and explanations under incomplete knowledge. In *Proceedings of AAAI'13*, M. desJardins and M. Littman, Eds. AAAI Press, 209–216.

Eiter, T., Faber, W., Leone, N., Pfeifer, G., and Polleres, A. 2003. A logic programming approach to knowledge-state planning. *Artificial Intelligence 144,* 1-2, 157–211.

Fandinno, J., Mishra, S., Romero, J., and Schaub, T. 2020. Answer set programming made easy. In *Proceedings of ASPOCP'20*, M. Hecher and J. Zangari, Eds.

Gelfond, M. and Lifschitz, V. 1998. Action languages. *Electronic Transactions on Artificial Intelligence 3,* 6, 193–210.

Giunchiglia, E., Marin, P., and Narizzano, M. 2009. Reasoning with quantified Boolean formulas. In *Handbook of Satisfiability*, A. Biere, M. Heule, H. van Maaren, and T. Walsh, Eds. Frontiers in Artificial Intelligence and Applications, vol. 185. IOS Press, Chapter 24, 761–780.

Janhunen, T. 2004. Representing normal programs with clauses. In *Proceedings of ECAI'04*, R. López de Mántaras and L. Saitta, Eds. IOS Press, 358–362.

Janota, M. and Marques-Silva, J. 2015. Solving QBF by clause selection. In *Proceedings of IJCAI'15*, Q. Yang and M. Wooldridge, Eds. AAAI Press, 325–331.

Lifschitz, V. 2002. Answer set programming and plan generation. *Artificial Intelligence 138,* 1-2, 39–54.

Lifschitz, V. and Turner, H. 1994. Splitting a logic program. In *Proceedings of the Eleventh International Conference on Logic Programming*. MIT Press, 23–37.

Lonsing, F. and Egly, U. 2017. Depqbf 6.0: A search-based QBF solver beyond traditional QCDCL. In *Proceedings of CADE'17*. Springer-Verlag, 371–384.

Lonsing, F. and Egly, U. 2019. Qratpre+: Effective QBF preprocessing via strong redundancy properties. In *Proceedings of SAT'19*. Springer-Verlag, 203–210.

Mayer-Eichberger, V. and Saffidine, A. 2020. Positional games and QBF: the corrective encoding. In *Proceedings of SAT'20*. Springer-Verlag, 447–463.

Niemelä, I. 2008. Answer set programming without unstratified negation. In *Proceedings of ICLP'08*. Springer-Verlag, 88–92.

Palacios, H. and Geffner, H. 2005. Mapping conformant planning into SAT through compilation and projection. In *Proceedings of CAEPIA'05*. Springer-Verlag, 311–320.

Palacios, H. and Geffner, H. 2009. Compiling uncertainty away in conformant planning problems with bounded width. *Journal of Artificial Intelligence Research 35*, 623–675.

Peitl, T., Slivovsky, F., and Szeider, S. 2019. Dependency learning for QBF. *Journal of Artificial Intelligence Research 65*, 180–208.

Rabe, M. N. and Tentrup, L. 2015. CAQE: A certifying QBF solver. In *Proceedings of FMCAD'15*, R. Kaivola and T. Wahl, Eds. IEEE Computer Society Press, 136–143.

Rintanen, J. 1999. Constructing conditional plans by a theorem-prover. *Journal of Artificial Intelligence Research 10*, 323–352.

Son, T., Tu, P., Gelfond, M., and Morales, A. 2005. An approximation of action theories of $\mathcal{AL}$ and its application to conformant planning. In *Proceedings of LPNMR'05*. Springer-Verlag, 172–184.

Tu, P., Son, T., and Baral, C. 2007. Reasoning and planning with sensing actions, incomplete information, and static causal laws using answer set programming. *Theory and Practice of Logic Programming 7,* 4, 377–450.

Tu, P., Son, T., Gelfond, M., and Morales, A. 2011. Approximation of action theories and its application to conformant planning. *Artificial Intelligence 175,* 1, 79–119.

Turner, H. 2002. Polynomial-length planning spans the polynomial hierarchy. In *Proceedings of JELIA'02*. Springer-Verlag, 111–124.

Wimmer, R., Reimer, S., Marin, P., and Becker, B. 2017. Hqspre – an effective preprocessor for QBF and DQBF. In *Proceedings of TACAS'17*. Springer-Verlag, 373–390.

Yalciner, I., Nouman, A., Patoglu, V., and Erdem, E. 2017. Hybrid conditional planning using answer set programming. *Theory and Practice of Logic Programming 17,* 5-6, 1027–1047.

Appendix A Results

In this section, we provide some additional information about the experiments, and present the results for every solver and configuration.

We note that in conditional planning, CPasp receives as input an additional width parameter that determines the minimum number of branches of a conditional plan. qasp2qbf receives no information of that kind, and in this regard it solves a more difficult task than CPasp. On the other hand, that width parameter allows CPasp to return the complete conditional plan, while qasp2qbf only returns the assignment to the first action of the plan. Observe also that given that CPasp is incomplete, in some instances qasp2qbf can find plans for opt whose length is smaller than the fixed length of sat. This explains why in some few cases qasp2qbf solves faster the opt problem than sat.

In the experiments, the times for grounding and for the translators lp2normal and lp2sat are negligible, while, in the sat problem, preprocessing takes on average 5 and 2 seconds in conformant and conditional planning, respectively. We expect these times to be similar for the opt problem. The reported times are dominated by clingo's solving time in CPasp, and by the QBF solvers in qasp2qbf. But the usage of preprocessors is fundamental for the performance of qasp2qbf. For example, in conformant planning, caqe alone results in 18 timeouts, while with qratpre+ they go down to only 6.

	clingo^{C}		clingo^{F}		clingo^{H}		clingo^{J}		clingo^{R}		clingo^{T}	
Bt	52 (0)	0	176 (0)	0	362 (1)	0	43 (0)	0	363 (1)	0	48 (0)	0
Bmt	49 (0)	0	163 (0)	0	238 (0)	0	50 (0)	0	363 (1)	0	35 (0)	0
Btc	236 (0)	0	395 (1)	0	375 (1)	0	258 (0)	0	380 (1)	0	282 (0)	0
Bmtc	746 (2)	211	998 (2)	0	743 (2)	0	744 (2)	0	773 (2)	0	739 (2)	0
Btuc	367 (1)	0	389 (1)	0	372 (1)	0	245 (0)	0	382 (1)	0	380 (1)	0
Bmtuc	749 (2)	0	938 (2)	0	741 (2)	0	739 (2)	0	778 (2)	0	742 (2)	0
Domino	0 (0)	0	0 (0)	0	0 (0)	0	0 (0)	0	0 (0)	0	0 (0)	0
Ring	774 (2)	506	1102 (3)	777	538 (1)	96	673 (1)	416	645 (1)	204	803 (2)	364
Ringu	1800 (5)	0	1800 (5)	0	1800 (5)	0	1800 (5)	0	1800 (5)	0	1800 (5)	0
Total	530 (12)	79	662 (14)	86	574 (13)	10	505 (10)	46	609 (14)	22	536 (12)	40

Table A 1: Conformant planning with clingo

	caqe$^{\circ}$		caqe^{B}		caqe^{H}		caqe^{Q}	
Bt	69 (0)	3	377 (0)	1	364 (1)	1	46 (0)	0
Bmt	44 (0)	3	149 (0)	4	373 (1)	11	155 (0)	2
Btc	440 (1)	3	487 (1)	3	577 (1)	13	413 (1)	1
Bmtc	814 (2)	7	777 (2)	7	1083 (3)	32	771 (2)	9
Btuc	476 (1)	3	475 (1)	3	606 (1)	9	400 (1)	1
Bmtuc	849 (2)	14	789 (2)	15	1260 (3)	25	780 (2)	14
Domino	1500 (5)	1500	0 (0)	0	0 (0)	0	0 (0)	0
Ring	1104 (3)	1097	219 (0)	30	837 (2)	793	281 (0)	40
Ringu	1630 (4)	1643	1018 (2)	969	1506 (4)	1505	246 (0)	45
Total	769 (18)	474	476 (8)	114	734 (16)	265	343 (6)	12

Table A 2: Conformant planning with caqe

	DEPQBF$^{\circ}$		DEPQBF^{B}		DEPQBF^{H}		DEPQBF^{Q}	
Bt	740 (2)	0	744 (2)	1	363 (1)	1	319 (0)	0
Bmt	974 (2)	2	1089 (3)	3	372 (1)	11	367 (1)	2
Btc	1111 (3)	3	1084 (3)	3	729 (2)	10	566 (1)	2
Bmtc	1494 (4)	5	1442 (4)	7	1082 (3)	20	1083 (3)	9
Btuc	1102 (3)	7	1088 (3)	3	730 (2)	11	538 (1)	1
Bmtuc	1479 (4)	82	1443 (4)	8	1440 (4)	39	1085 (3)	12
Domino	0 (0)	0	1 (0)	0	0 (0)	0	0 (0)	0
Ring	1800 (5)	1800	1708 (4)	1468	1108 (3)	736	1120 (3)	1109
Ringu	1800 (5)	1800	1800 (5)	1800	1800 (5)	1800	1800 (5)	1800
Total	1166 (28)	411	1155 (28)	365	847 (21)	292	764 (17)	326

Table A 3: Conformant planning with DEPQBF

	QESTO$^{\circ}$		QESTO^{B}		QESTO^{H}		QESTO^{Q}	
Bt	725 (2)	3	458 (1)	1	363 (1)	1	57 (0)	0
Bmt	1081 (3)	9	443 (1)	4	372 (1)	13	76 (0)	1
Btc	1081 (3)	7	777 (2)	6	729 (2)	9	394 (1)	1
Bmtc	1440 (4)	20	1104 (3)	23	1082 (3)	23	826 (2)	9
Btuc	1082 (3)	15	736 (1)	6	734 (2)	9	391 (1)	2
Bmtuc	1441 (4)	359	1094 (3)	27	1286 (3)	26	813 (2)	12
Domino	1319 (4)	1339	0 (0)	0	0 (0)	0	0 (0)	0
Ring	1105 (3)	1096	499 (1)	30	782 (2)	752	790 (2)	392
Ringu	1800 (5)	1800	1444 (4)	1440	1800 (5)	1800	1800 (5)	1800
Total	1230 (31)	516	728 (16)	170	794 (19)	292	571 (13)	246

Table A 4: Conformant planning with QESTO

	QUTE$^{\circ}$		QUTE^{B}		QUTE^{H}		QUTE^{Q}	
Bt	864 (2)	3	1081 (3)	1	363 (1)	42	852 (2)	0
Bmt	1095 (3)	9	1180 (3)	3	382 (1)	12	477 (1)	1
Btc	1095 (3)	16	1093 (3)	3	729 (2)	38	1093 (3)	1
Bmtc	1453 (4)	27	1444 (4)	8	1082 (3)	374	1484 (4)	9
Btuc	1111 (3)	55	1101 (3)	3	731 (2)	38	1116 (3)	1
Bmtuc	1501 (4)	557	1745 (4)	24	1440 (4)	378	1483 (4)	12
Domino	1238 (4)	1239	0 (0)	0	0 (0)	0	0 (0)	0
Ring	1800 (5)	1449	1702 (4)	1441	1441 (4)	1440	1460 (4)	1208
Ringu	1800 (5)	1800	1800 (5)	1599	1800 (5)	1800	1800 (5)	1800
Total	1328 (33)	572	1238 (29)	342	885 (22)	458	1085 (26)	336

Table A 5: Conformant planning with QUTE

	CLINGO^{C}		CLINGO^{F}		CLINGO^{H}		CLINGO^{J}		CLINGO^{R}		CLINGO^{T}	
Bts1	1041 (2)	121	1080 (3)	77	795 (2)	49	989 (2)	51	916 (2)	28	927 (2)	370
Bts2	1080 (3)	60	1080 (3)	361	787 (2)	26	986 (2)	30	920 (2)	66	1080 (3)	71
Bts3	993 (2)	85	1071 (2)	215	802 (2)	82	1012 (2)	50	944 (2)	68	1080 (3)	253
Bts4	1060 (2)	69	1080 (3)	115	833 (2)	76	882 (2)	102	904 (2)	70	1080 (3)	173
Domino	872 (2)	15	885 (2)	59	906 (2)	6	771 (2)	13	786 (2)	7	759 (2)	18
Med	0 (0)	0	0 (0)	0	0 (0)	0	0 (0)	0	0 (0)	0	0 (0)	0
Ring	902 (2)	900	905 (2)	900	902 (2)	900	903 (2)	900	902 (2)	900	902 (2)	900
Sick	0 (0)	0	0 (0)	0	0 (0)	0	0 (0)	0	0 (0)	0	0 (0)	0
Total	743 (13)	156	762 (15)	215	628 (12)	142	692 (12)	143	671 (12)	142	728 (15)	223

Table A 6: Conditional planning with CLINGO

	CAQE$^{\circ}$		CAQE^{B}		CAQE^{H}		CAQE^{Q}	
Bts1	370 (1)	368	14 (0)	3	16 (0)	3	366 (1)	363
Bts2	378 (1)	376	14 (0)	3	15 (0)	4	106 (0)	54
Bts3	372 (1)	379	15 (0)	4	19 (0)	7	135 (0)	65
Bts4	374 (1)	384	17 (0)	4	25 (0)	8	107 (0)	53
Domino	372 (1)	365	362 (1)	125	366 (1)	297	223 (0)	113
Med	1095 (3)	1440	3 (0)	1	2 (0)	0	4 (0)	2
Ring	1350 (3)	1350	902 (2)	640	901 (2)	901	904 (2)	901
Sick	729 (2)	729	1 (0)	0	1 (0)	1	64 (0)	63
Total	630 (13)	673	166 (3)	97	168 (3)	152	238 (3)	201

Table A 7: Conditional planning with CAQE

	DEPQBF$^{\circ}$		DEPQBF^{B}		DEPQBF^{H}		DEPQBF^{Q}	
Bts1	722 (2)	720	364 (1)	361	396 (1)	366	721 (2)	506
Bts2	724 (2)	720	369 (1)	41	489 (1)	362	723 (2)	720
Bts3	729 (2)	720	372 (1)	10	721 (2)	361	726 (2)	720
Bts4	745 (2)	720	368 (1)	362	722 (2)	654	731 (2)	720
Domino	474 (1)	158	361 (1)	48	366 (1)	7	376 (1)	360
Med	2 (0)	0	3 (0)	1	2 (0)	0	1 (0)	0
Ring	1340 (2)	1350	901 (2)	900	902 (2)	901	902 (2)	902
Sick	55 (0)	48	1 (0)	0	1 (0)	0	0 (0)	0
Total	598 (11)	554	342 (7)	215	449 (9)	331	522 (11)	491

Table A 8: Conditional planning with DEPQBF

	QESTO$^{\circ}$		QESTO^{B}		QESTO^{H}		QESTO^{Q}	
Bts1	380 (1)	373	102 (0)	65	369 (1)	42	400 (1)	391
Bts2	384 (1)	379	370 (1)	182	720 (2)	720	390 (1)	378
Bts3	390 (1)	379	371 (1)	364	722 (2)	362	396 (1)	377
Bts4	392 (1)	382	379 (1)	89	721 (2)	720	392 (1)	379
Domino	720 (2)	413	363 (1)	5	366 (1)	360	530 (1)	481
Med	1100 (3)	1440	3 (0)	1	1 (0)	0	5 (0)	6
Ring	1350 (3)	1350	902 (2)	900	902 (2)	900	905 (2)	902
Sick	739 (2)	736	1 (0)	1	1 (0)	0	11 (0)	11
Total	681 (14)	681	311 (6)	200	475 (10)	388	378 (7)	365

Table A 9: Conditional planning with QESTO

	QUTE$^{\circ}$		QUTE^{B}		QUTE^{H}		QUTE^{Q}	
Bts1	770 (2)	760	365 (1)	361	724 (2)	722	751 (2)	726
Bts2	1080 (3)	1080	376 (1)	368	724 (2)	721	1080 (3)	1080
Bts3	896 (2)	901	380 (1)	370	783 (2)	777	1080 (3)	944
Bts4	966 (2)	916	390 (1)	377	731 (2)	726	1081 (3)	1080
Domino	722 (2)	408	361 (1)	360	720 (2)	720	606 (1)	451
Med	1104 (3)	1440	3 (0)	1	5 (0)	3	27 (0)	24
Ring	1350 (3)	1350	901 (2)	900	907 (2)	903	917 (2)	911
Sick	782 (2)	776	1 (0)	0	1 (0)	1	6 (0)	6
Total	958 (19)	953	347 (7)	342	574 (12)	571	693 (14)	652

Table A 10: Conditional planning with QUTE

Appendix B ASP(Q) and QASP

An ASP(Q) program Π is an expression of the form:

$$\Box_0 P_0 \Box_1 P_1 \cdots \Box_n P_n : C, \tag{B1}$$

where $n \geq 0$, $P_0, \ldots, P_n$ are logic programs, every $\Box_i$ is either $\exists^{st}$ or $\forall^{st}$, and C is a normal stratified program. In (Amendola et al. 2019), the logic programs P_i can be disjunctive. Here, for simplicity, we restrict ourselves to logic programs as defined in the Background section. Symbols $\exists^{st}$ and $\forall^{st}$ are called existential and universal answer set quantifiers, respectively. We say that the program P_i is existentially (universally, respectively) quantified if $\Box_i$ is $\exists^{st}$ ($\forall^{st}$, respectively). Let X and Y be sets of atoms such that $X \subseteq Y$, by $\mathit{fixfact}(X, Y)$ we denote the set of rules $\{x \leftarrow \mid x \in X\} \cup \{\leftarrow x \mid x \in Y \setminus X\}$. Let $\mathit{atoms}(P)$ denote the set of atoms occurring in a logic program P. Given an ASP(Q) program Π, a logic program P, and a set of atoms X, we denote by $\Pi_{P,X}$ the program of the form (B1), where P_0 is replaced by $P_0 \cup \mathit{fixfact}(X, \mathit{atoms}(P))$, that is, $\Pi_{P,X} = \Box_0(P_0 \cup \mathit{fixfact}(X, \mathit{atoms}(P))) \cdots \Box_n P_n : C$. The coherence of ASP(Q) programs can be defined recursively as follows:

- $\exists^{st} P : C$ is coherent, if there exists $M \in SM(P)$ such that $C \cup \mathit{fixfact}(M, \mathit{atoms}(P))$ is satisfiable;
- $\forall^{st} P : C$ is coherent, if for every $M \in SM(P)$, $C \cup \mathit{fixfact}(M, \mathit{atoms}(P))$ is satisfiable;
- $\exists^{st} P \Pi$ is coherent, if there exists $M \in SM(P)$ such that $\Pi_{P,M}$ is coherent.
- $\forall^{st} P \Pi$ is coherent, if for every $M \in SM(P)$, $\Pi_{P,M}$ is coherent.

Before moving to the translations between ASP(Q) and QASP, we introduce some special forms of ASP(Q) programs. We say that an ASP(Q) program of the form (B1) is in normal form if C is empty and the heads of the rules of every P_i do not contain atoms occurring in any P_j such that $j < i$, and we say that an ASP(Q) program is in $\forall$-GDT form if all its universally quantified logic programs are in GDT form. ASP(Q) programs of the form (B1) can be translated to ASP(Q) programs in normal and $\forall$-GDT form (using auxiliary variables) following the next steps:

1. Replace program C by $\emptyset$, and add $\exists^{st} C$ after $\Box_n P_n$.
2. In every logic program, replace the normal rules $p \leftarrow B$ such that p occurs in some previous program by constraints $\bot \leftarrow \neg p, B$; and erase the choice rules $\{p\} \leftarrow B$ such that p occurs in some previous program.
3. Apply the method described in (Niemelä 2008; Fandinno et al. 2020) to translate the universally quantified programs to GDT form.

The translation can be computed in polynomial time on the size of the ASP(Q) program, and the result is an ASP(Q) program in normal and $\forall$-GDT form that is coherent if and only if the original ASP(Q) program is coherent.

Once an ASP(Q) program Π of the form (B1) is in normal and $\forall$-GDT form, we can translate it to a QLP as follows. Let $qlp(\Pi)$ be the QLP of the form

$$Q_0 X_0 Q_1 X_1 \ldots Q_n X_n \big(P_0^\bullet \cup P_1^\bullet \cup \ldots \cup P_n^\bullet \cup O\big)$$

where for every $i \in \{0, \ldots, n\}$, if $\Box_i$ is $\exists^{st}$ then Q_i is $\exists$, X_i is the set of atoms occurring in P_i that do not occur in any previous program, and $P_i^\bullet$ is P_0 if $n = 0$ and otherwise it is the program that results from adding the literal $\neg\alpha_i$ to the body of every rule of P_i; while if $\Box_i$ is $\forall^{st}$ then Q_i is $\forall$, X_i is $\{p \mid \{p\} \leftarrow \in P_i\}$, and $P_i^\bullet$ is the program that results from replacing $\bot$ in the head of every integrity constraint by α_i; and O is $\{\alpha_i \leftarrow \alpha_{i-1} \mid \beta \leq i \leq n\}$ where β is 1 if $\Box_0 = \forall^{st}$ and is 2 otherwise.

Theorem Appendix B.1
Π is coherent if and only if $qlp(\Pi)$ is satisfiable.

The translation in the other direction is as follows. Given a QLP $\mathcal{Q}P$ over $\mathcal{A}$ of the form (2), let $aspq(\mathcal{Q}P)$ be the ASP(Q) program of the form

$$Q_0^{st} P_0 Q_1^{st} P_1 \ldots Q_n^{st} P_n \exists^{st} (P \cup O) : \emptyset$$

where every P_i is $\{\{p'\} \leftarrow \mid p \in X_i\}$, O is $\{\bot \leftarrow p, \neg p' \mid p \in \mathcal{X}\} \cup \{\bot \leftarrow \neg p, p' \mid p \in \mathcal{X}\}$ where $\mathcal{X} = \bigcup_{i \in \{0,\ldots,n\}} X_i$, and we assume that the set $\{p' \mid p \in \mathcal{X}\}$ is disjunct from $\mathcal{A}$.

Theorem Appendix B.2
$\mathcal{Q}P$ is satisfiable if and only if $aspq(\mathcal{Q}P)$ is coherent.

Appendix C Proof of Theorem 6.1

We first introduce some notation. Let $\mathcal{D}$ be a planning description. By $C_{\mathcal{D}}$ we denote the logic program $\mathit{Domain} \cup \mathcal{D}^\bullet \cup (3) \cup (5)$. Then (6) can be rewritten as $\exists \mathit{Occ} \forall \mathit{Open}\ C_{\mathcal{D}}$. Let *Generate* denote (3), and *Alpha* denote (5). Then $C_{\mathcal{D}}$ can be rewritten as $\mathit{Domain} \cup \mathcal{D}^\bullet \cup \mathit{Generate} \cup \mathit{Alpha}$. Since all occurrences of variable T in $C_{\mathcal{D}}$ are always bound by an atom $t(T)$, and the only ground atoms of that form that can be true are $t(1), \ldots, t(n)$, we can consider only the ground instantiations of $C_{\mathcal{D}}$ where T has the values from 1 to n. We add some more notation to represent those ground instances. We denote by $r(i)$ the rule obtained by replacing in r the variable T by i. If $i \leq j$, by $r(i,j)$ we denote the set of rules containing $r(k)$ for all $i \leq k \leq j$. For a set of rules R, we define $R(i) = \{r(i) \mid r \in R\}$ and $R(i,j) = \{r(i,j) \mid r \in R\}$. Let $C_{\mathcal{D}}[i]$ denote the program

$$\mathit{Alpha}(i) \cup \mathit{Generate}(i) \cup \mathcal{D}^\bullet_{\mathbf{D}}(i)$$

and $C_{\mathcal{D}}[0,j]$ denote the program

$$\mathit{Domain} \cup \mathcal{D}^\bullet_{\mathbf{I}} \cup \bigcup_{i=1}^{j} C_{\mathcal{D}}[i].$$

Note that $C_{\mathcal{D}}$ has the same stable models as the program

$$C_{\mathcal{D}}[0,n] \cup \mathcal{D}^\bullet_{\mathbf{G}}$$

that is the same as

$$C_{\mathcal{D}}[0,j] \cup \bigcup_{i=j+1}^{n} C_{\mathcal{D}}[i] \cup \mathcal{D}^\bullet_{\mathbf{G}}$$

for every $j \in \{1, \ldots, n\}$. The following results make repetitive use of the Splitting Theorem (Lifschitz and Turner 1994). Note that no atom occurring in $C_{\mathcal{D}}[0,j]$ occurs in the head of a rule in $C_{\mathcal{D}}[i] \cup \mathcal{D}^\bullet_{\mathbf{G}}$ for any $i > j$. We also define $\mathit{Holds}[i] = \{h(f,i) \mid f \in F\}$.

Lemma 1
Let $\mathcal{D}$ be a planning description that represents a conformant planning problem $\mathcal{PP}$, $s, s' \subseteq F$ be two states, a be an action, and i be a positive integer. Moreover, let X be a stable model of $C_{\mathcal{D}}[0, i-1]$ such that $s' = \{f \in F \mid h(f, i-1) \in X\}$ and $\alpha(i-1) \notin X$. Then, $\Phi(s', a) = s$ iff there is a unique stable model Y of

$$X \cup C_{\mathcal{D}}[i] \cup \mathit{fixcons}(\{\mathit{occ}(a,i)\}, \mathit{Occ}_i) \tag{C1}$$

such that $s = \{f \in F \mid h(f,i) \in Y\}$ and $\alpha(i) \notin Y$. Furthermore, if $\Phi(s', a) = \bot$ then (C1) is unsatisfiable.

Proof
Let

$$P_1 = X \cup \mathit{Alpha}(i) \cup \mathit{Generate}(i) \cup \mathit{fixcons}(\{\mathit{occ}(a,i)\}, \mathit{Occ}_i)$$
$$P_2 = P_1 \cup \mathcal{D}^\bullet_{\mathbf{D}}(i)$$

Recall that $C_{\mathcal{D}}[i] = \mathit{Alpha}(i) \cup \mathit{Generate}(i) \cup \mathcal{D}^\bullet_{\mathbf{D}}(i)$ and, thus, P_2 is equal to (C1). Program P_1 has a unique stable model $X_1 = X \cup \{\mathit{occ}(a,i)\}$ and, by the Splitting Theorem, the stable models of P_2 are the same as the stable models of $X_1 \cup \mathcal{D}^\bullet_{\mathbf{D}}(i)$. Note that $\alpha(i-1) \notin X$ implies that $\alpha(i) \notin X_1$ because the only rule with $\alpha(i)$ in the head has $\alpha(i-1)$ in the body. Furthermore, since $\mathcal{D}$ represents $\mathcal{PP}$ and $\alpha(i) \notin X$, it follows that

$$(X \cap Holds[i-1]) \cup \{occ(a,i)\} \cup \{t(i)\} \cup \mathcal{D}^{\bullet}_{\mathbf{D}}(i) \tag{C2}$$

either (Case 1) has a unique stable model Y' such that $s = \{f \in F \mid h(f,i) \in Y'\}$ or (Case 2) has no stable model and $\Phi(s', a) = \bot$. Recall that $\mathbf{D}(\mathcal{D})$ is deterministic. Hence, it cannot generate multiple stable models, and the same applies to $\mathcal{D}^{\bullet}_{\mathbf{D}}(i)$. Note also that $\alpha(i) \notin X$ implies $\alpha(i) \notin Y'$ because the only rules with $\alpha(i)$ in the head belong to P_1. In Case 1 the iff holds because $Y = Y' \cup X$ is the unique stable model of $X_1 \cup \mathcal{D}^{\bullet}_{\mathbf{D}}(i)$, and it satisfies the conditions $s = \{f \in F \mid h(f,i) \in Y\}$ and $\alpha(i) \notin Y$. To see this, observe that the atoms in X_1 that do not belong to (C2) also do not occur in the rules of $\mathcal{D}^{\bullet}_{\mathbf{D}}(i)$. In Case 2 the iff holds because none of its two sides holds. Finally, if $\Phi(s', a) = \bot$ then (C2) is unsatisfiable, hence $X_1 \cup \mathcal{D}^{\bullet}_{\mathbf{D}}(i)$ is also unsatisfiable, and therefore (C1) is unsatisfiable. □

Lemma 2

Let $\mathcal{D}$ be a planning description that represents a conformant planning problem $\mathcal{PP}$, $s, s' \subseteq F$ be two states, $p = [a_1; a_2; \dots ; a_n]$ be a plan, i be a positive integer, and X be a stable model of $C_{\mathcal{D}}[0, i-1]$ such that $s' = \{f \in F \mid h(f, i-1) \in X\}$ and $\alpha(i-1) \notin X$. Moreover, let

$$A = \{occ(a_1, i), occ(a_2, i+1), \dots, occ(a_n, i+n-1)\}$$

be a set of atoms representing action occurrences. Then, $\Phi(\{s'\}, p) = s$ iff there is a unique stable model Y of

$$X \cup \bigcup_{j=i}^{i+n-1} \big(C_{\mathcal{D}}[j] \cup fixcons(\{occ(a,j)\}, Occ_j)\big) \tag{C3}$$

such that $s = \{f \in F \mid h(f, i+n-1) \in Y\}$. Furthermore, $\alpha(i+n-1) \notin Y$.

Proof

We consider first the case where $\Phi(s', a_1)$ is $\bot$, and then the case where $\Phi(s', a_1)$ is a state s''. If $\Phi(s', a_1)$ is $\bot$, then by Lemma 1 the program

$$X \cup C_{\mathcal{D}}[i] \cup fixcons(\{occ(a,i)\}, Occ_i) \tag{C4}$$

is unsatisfiable. Hence, by the Splitting Theorem, the program (C3) is also unsatisfiable, and the iff holds because none of its sides holds. We continue with the case where $\Phi(s', a_1)$ is a state s''. If $n = 1$, the lemma statement follows directly from Lemma 1. Then the proof follows by induction assuming that the lemma statement holds for all plans of length $n-1$. By definition, $\Phi(\{s'\}, p) = \Phi(\{s''\}, q)$ with $q = [a_2; \dots ; a_n]$. From Lemma 1, we get that $\Phi(s', a_1) = s''$ iff there is a unique stable model Z of (C4) such that $s'' = \{f \in F \mid h(f,i) \in Z\}$ and $\alpha(i) \notin Z$. By induction hypothesis, we get that $\Phi(\{s''\}, q) = s$ iff there is a unique stable model Y of

$$Z \cup \bigcup_{j=i+1}^{i+n-1} \big(C_{\mathcal{D}}[j] \cup fixcons(\{occ(a,j)\}, Occ_j)\big)$$

such that $s = \{f \in F \mid h(f, i+n-1) \in Y\}$. Furthermore, $\alpha(i+n-1) \notin Y$. From the Splitting Theorem, Y is the unique stable model of (C3), and the iff condition holds. $\square$

Lemma 3

Let $\mathcal{D}$ be a planning description that represents a conformant planning problem $\mathcal{PP}$. Let $Y \subseteq \mathit{Open}$ be a set of atoms, $s \subseteq F$ and $s_0 \in SM(\mathbf{I}(\mathcal{D}))$ be states such that

$$Y = \{h(f,0) \in \mathit{Open} \mid f \in s_0\},$$

and $p = [a_1; a_2; \dots; a_n]$ be a plan with

$$X = \{\mathit{occ}(a_1,1), \mathit{occ}(a_2,2), \dots, \mathit{occ}(a_n,n)\}.$$

Then, $\Phi(\{s_0\}, p) = s$ iff there is a unique stable model Z of

$$C_{\mathcal{D}}[0,n] \cup \mathit{fixcons}(X, \mathit{Occ}) \cup \mathit{fixcons}(Y, \mathit{Open}) \tag{C5}$$

such that $s = \{f \mid h(f,n) \in Z\}$. Furthermore, $\alpha(n) \notin Z$.

Proof

Note that $C_{\mathcal{D}}[0,0] = \mathit{Domain} \cup \mathcal{D}^{\bullet}_{\mathbf{I}}$ and that $s_0 \in SM(\mathbf{I}(\mathcal{D}))$ implies that this program has a unique stable model Z_0 such that $s_0 = \{f \in F \mid h(f,0) \in Z_0\}$ and $\alpha(0) \notin Z_0$. Then, from Lemma 2 with $i = 1$, we get that $\Phi(\{s_0\}, p) = s$ iff there is a unique stable model Z of

$$Z_0 \cup \bigcup_{j=1}^{n} \big(C_{\mathcal{D}}[j] \cup fixcons(\{\mathit{occ}(a,j)\}, \mathit{Occ}_j)\big).$$

Furthermore, $\alpha(n) \notin Z$. From the Splitting Theorem, this implies that Z is a stable model of $C_{\mathcal{D}}[0,n]$. Furthermore, it is the unique stable model of $C_{\mathcal{D}}[0,n]$ such that

$$s_0 = \{f \in F \mid h(f,0) \in Z_0\} = \{f \in F \mid h(f,0) \in Z\}.$$

Note also that $Y = \{h(f,0) \in \mathit{Open} \mid f \in s_0\} = Z \cap \mathit{Open}$ and, thus, Z satisfies

$$\mathit{fixcons}(Y, \mathit{Open}) = \mathit{fixcons}(Z \cap \mathit{Open}, \mathit{Open}).$$

No other stable model of $C_{\mathcal{D}}[0,0]$ satisfies these constraints. Therefore, Z is the unique stable model of (C5). $\square$

Lemma 4

Let $\mathcal{D}$ be a planning description that represents a conformant planning problem $\mathcal{PP}$ such that $\mathbf{I}(\mathcal{D})$ is in *GDT* form, and n be a positive integer. Let $p = [a_1; \dots; a_n]$ be a plan, $X = \{\mathit{occ}(a_1,1), \dots \mathit{occ}(a_n,n)\}$ be a set of atoms, and $Y \subseteq \mathit{Open}$ be another set of atoms. Then, the following two statements are equivalent:

- there is some stable model Z of $C_{\mathcal{D}} \cup \mathit{fixcons}(X, \mathit{Occ}) \cup \mathit{fixcons}(Y, \mathit{Open})$ such that $\alpha(n) \notin Z$; and
- there is $s_0 \in SM(\mathbf{I}(\mathcal{D}))$ s.t. $Y = \{h(f,0) \in \mathit{Open} \mid f \in s_0\}$ and $\Phi_{\mathbf{D}(\mathcal{D})}(\{s_0\}, p) \subseteq G$.

Proof

First, assume that there is no $s_0 \in SM(\mathbf{I}(\mathcal{D}))$ s.t. $Y = \{h(f,0) \in \mathit{Open} \mid f \in s_0\}$. In this case, the second lemma statement does not hold, and we will see next that the first lemma statement does not hold either. Given our assumption and by construction, the program $\mathit{Domain} \cup \mathcal{D}^{\bullet}_{\mathbf{I}} \cup \mathit{Alpha} \cup \mathit{fixcons}(Y, \mathit{Open})$, that is a subset of $C_{\mathcal{D}} \cup \mathit{fixcons}(X, \mathit{Occ}) \cup \mathit{fixcons}(Y, \mathit{Open})$, has a unique stable model, and this stable model contains the atom $\alpha(n)$. Then, by the Splitting Theorem, the stable models of $C_{\mathcal{D}} \cup \mathit{fixcons}(X, \mathit{Occ}) \cup \mathit{fixcons}(Y, \mathit{Open})$ contain the atom $\alpha(n)$, which contradicts the first lemma statement.

Otherwise, there is $s_0 \in SM(\mathbf{I}(\mathcal{D}))$ such that $Y = \{h(f,0) \in \mathit{Open} \mid f \in s_0\}$. Then, from Lemma 3, it follows that $\Phi_{\mathbf{D}(\mathcal{D})}(\{s_0\}, p) = s$ iff there is a unique stable model Z of

$$C_{\mathcal{D}}[0,n] \cup \mathit{fixcons}(X, \mathit{Occ}) \cup \mathit{fixcons}(Y, \mathit{Open})$$

such that $s = \{f \in F \mid h(f,n) \in Z\}$. Furthermore, $\alpha(n) \notin Z$. Note that $C_{\mathcal{D}} = C_{\mathcal{D}}[0,n]$. Moreover, we can see that $s \in G$ iff Z satisfies $\mathcal{D}^{\bullet}_{\mathbf{G}}$. Since $\mathcal{D}^{\bullet}_{\mathbf{G}} \subseteq \mathcal{D}^{\bullet}$, it follows that the two lemma statements are equivalent. □

Lemma 5

Let $\mathcal{D}$ be a planning description that represents a conformant planning problem $\mathcal{PP}$ such that $\mathbf{I}(\mathcal{D})$ is in *GDT* form. Then, the following statements hold:

- For every $Y \subseteq \mathit{Open}$ such that $\mathcal{D}^{\circ}_{\mathbf{I}} \cup \mathit{fixcons}(Y, \mathit{Open})$ is satisfiable, there is a unique $s_Y \in SM(\mathbf{I}(\mathcal{D}))$ such that $Y = \{h(f,0) \in \mathit{Open} \mid f \in s_Y\}$.
- For every $s_0 \in SM(\mathbf{I}(\mathcal{D}))$, there is a unique $Y \subseteq \mathit{Open}$ such that $Y = \{h(f,0) \in \mathit{Open} \mid f \in s_0\}$. Furthermore, $\mathcal{D}^{\circ}_{\mathbf{I}} \cup \mathit{fixcons}(Y, \mathit{Open})$ is satisfiable.

Proof

For the first statement, note that there is a choice rule of form $\{f\} \leftarrow$ in $\mathbf{I}(\mathcal{D})$ iff the atom $h(f,0)$ belongs to *Open*. The rest of the program is deterministic, therefore, there is at most one $s_0 \in SM(\mathbf{I}(\mathcal{D}))$ such that $Y = \{h(f,0) \in \mathit{Open} \mid f \in s_0\}$. Furthermore, since $\mathcal{D}^{\circ}_{\mathbf{I}} \cup \mathit{fixcons}(Y, \mathit{Open})$ is satisfiable, such stable model exists. For the same reason, for the second statement, if $s_0 \in SM(\mathbf{I}(\mathcal{D}))$, there is a unique $Y \subseteq \mathit{Open}$ such that $Y = \{h(f,0) \in \mathit{Open} \mid f \in s_0\}$. Furthermore, it is easy to see that $\mathcal{D}^{\circ}_{\mathbf{I}} \cup \mathit{fixcons}(Y, \mathit{Open})$ is satisfiable. □

Lemma 6

Let $\mathcal{D}$ be a planning description that represents a conformant planning problem $\mathcal{PP}$ such that $\mathbf{I}(\mathcal{D})$ is in *GDT* form, and n be a positive integer. Let $p = [a_1; \ldots; a_n]$ be a plan, and $X = \{occ(a_1, 1), \ldots occ(a_n, n)\}$ be a set of atoms. Then, the following two statements are equivalent:

- for all $Y \subseteq \mathit{Open}$ such that $\mathcal{D}^{\circ}_{\mathbf{I}} \cup \mathit{fixcons}(Y, \mathit{Open})$ is satisfiable, there is some stable model Z of $C_{\mathcal{D}} \cup \mathit{fixcons}(X, \mathit{Occ}) \cup \mathit{fixcons}(Y, \mathit{Open}))$ such that $\alpha(n) \notin Z$; and
- every $s_0 \in SM(\mathbf{I}(\mathcal{D}))$ satisfies $\Phi_{\mathbf{D}(\mathcal{D})}(\{s_0\}, p) \subseteq G$.

Proof
From Lemma 4, the first lemma statement holds iff

> for all $Y \subseteq Open$ such that $\mathcal{D}^{\circ}_{\mathbf{I}} \cup \mathit{fixcons}(Y, Open)$ is satisfiable,
> there is $s \in SM(\mathbf{I}(\mathcal{D}))$ such that $Y = \{h(f, 0) \in Open \mid f \in s\}$
> and $\Phi_{\mathbf{D}(\mathcal{D})}(\{s\}, p) \subseteq G$.

Furthermore, for every $Y \subseteq Open$, there is at most one state $s_0 \in SM(\mathbf{I}(\mathcal{D}))$ such that $Y = \{h(f, 0) \in Open \mid f \in s_0\}$ (Lemma 5). Therefore, the above statement can be rewritten as

> every $Y \subseteq Open$ satisfies that, if $\mathcal{D}^{\circ}_{\mathbf{I}} \cup \mathit{fixcons}(Y, Open)$ is satisfiable,
> then $\Phi_{\mathbf{D}(\mathcal{D})}(\{s_Y\}, p) \subseteq G$ where $s_Y \in SM(\mathbf{I}(\mathcal{D}))$ is the unique state such (C6)
> that $Y = \{h(f, 0) \in Open \mid f \in s_Y\}$.

First-to-second statement. Pick any $s_0 \in SM(\mathbf{I}(\mathcal{D}))$. From Lemma 5, there is a unique set $Y \subseteq Open$ s.t. $Y = \{h(f, 0) \in Open \mid f \in s_0\}$. Furthermore, by construction, $\mathcal{D}^{\circ}_{\mathbf{I}} \cup \mathit{fixcons}(Y, Open)$ is satisfiable. From (C6), this implies $\Phi_{\mathbf{D}(\mathcal{D})}(\{s_0\}, p) \subseteq G$ and, thus, the second lemma statement holds.

Second-to-first statement. Pick any $Y \subseteq Open$ such that $\mathcal{D}^{\circ}_{\mathbf{I}} \cup \mathit{fixcons}(Y, Open)$ is satisfiable. From Lemma 5, this implies that there is a unique $s_Y \in SM(\mathbf{I}(\mathcal{D}))$ such that $Y = \{h(f, 0) \in Open \mid f \in s_Y\}$. From the second lemma statement, this implies that $\Phi_{\mathbf{D}(\mathcal{D})}(\{s_Y\}, p) \subseteq G$. Hence, (C6) and the first lemma statement hold. □

Lemma 7
Let $\mathcal{D}$ be a planning description that represents a conformant planning problem $\mathcal{PP}$ such that $\mathbf{I}(\mathcal{D})$ is in *GDT* form, and n be a positive integer. Let $p = [a_0; \dots ; a_n]$ be a plan, $X = \{occ(a_1, 1), \dots occ(a_n, n)\}$ be a set of atoms, and $Y \subseteq Open$. Then, the following two statements are equivalent:

- $C_{\mathcal{D}} \cup \mathit{fixcons}(X, Occ) \cup \mathit{fixcons}(Y, Open)$ is satisfiable; and
- if $\mathcal{D}^{\circ}_{\mathbf{I}} \cup \mathit{fixcons}(Y, Open)$ is satisfiable, then there is some stable model Z of $C_{\mathcal{D}} \cup \mathit{fixcons}(X, Occ) \cup \mathit{fixcons}(Y, Open))$ such that $\alpha(n) \notin Z$.

Proof
We proceed by cases. Assume first that $\mathcal{D}^{\circ}_{\mathbf{I}} \cup \mathit{fixcons}(Y, Open)$ is satisfiable. Then, the second lemma statement trivially implies the first. Next, assume that the first lemma statement holds. We show that the second lemma statement follows. By definition, the first lemma statement implies that there is some stable model Z of $C_{\mathcal{D}} \cup \mathit{fixcons}(X, Occ) \cup \mathit{fixcons}(Y, Open))$. Hence, we only need to show that $\alpha(n) \notin Z$ and, for this, it is enough to show that $\alpha(0) \notin Z$. Suppose, for the sake of contradiction, that $\alpha(0) \in Z$. From the Splitting Theorem, there is a stable model Z_0 of $\mathcal{D}^{\bullet}_{\mathbf{I}} \cup \mathit{fixcons}(Y, Open)$ such that $\alpha(0) \in Z_0$. By construction, this implies that $\mathcal{D}^{\circ}_{\mathbf{I}} \cup \mathit{fixcons}(Y, Open)$ is not satisfiable, which is a contradiction with the assumption.

Assume now that $\mathcal{D}^{\circ}_{\mathbf{I}} \cup \mathit{fixcons}(Y, Open)$ is not satisfiable. Then, the second lemma statement trivially holds. Furthermore, by construction, every stable model Z_0 of $\mathcal{D}^{\bullet}_{\mathbf{I}} \cup \mathit{fixcons}(Y, Open)$ satisfies $\alpha(0) \in Z_0$. Let $Z = Z_0 \cup X \cup \{\alpha(i) \mid 1 \leq i \leq n\}$. It holds that

- the bodies of all rules in $\mathcal{D}^{\bullet}_{\mathbf{D}} \cup \mathcal{D}^{\bullet}_{\mathbf{G}}$ are not satisfied by Z,

- the bodies of all rules in *Alpha* are satisfied by Z and all $\alpha(i)$ are thus supported, and
- all rules in *Generate* are satisfied by Z and all elements in X are supported.

Thus Z is a stable model of $C_{\mathcal{D}} \cup \mathit{fixcons}(X, \mathit{Occ}) \cup \mathit{fixcons}(Y, \mathit{Open})$ and the first lemma statement holds. □

Lemma 8
Let $\mathcal{D}$ be a planning description that represents a conformant planning problem $\mathcal{PP}$ such that $\mathbf{I}(\mathcal{D})$ is in *GDT* form, and n be a positive integer. Let $p = [a_0; \ldots ; a_n]$ be a plan and $X = \{occ(a_0, 0), \ldots occ(a_n, n)\}$ be a set of atoms. Then, the following two statements are equivalent:

- for all $Y \subseteq \mathit{Open}$, $C_{\mathcal{D}} \cup \mathit{fixcons}(X, \mathit{Occ}) \cup \mathit{fixcons}(Y, \mathit{Open})$ is satisfiable; and
- every $s_0 \in SM(\mathbf{I}(\mathcal{D}))$ satisfies $\Phi_{\mathbf{D}(\mathcal{D})}(\{s_0\}, p) \subseteq G$.

Proof
The first lemma statement holds
iff for all $Y \subseteq \mathit{Open}$ satisfying that $\mathcal{D}^{\circ}_{\mathbf{I}} \cup \mathit{fixcons}(Y, \mathit{Open})$ is satisfiable, there is some stable model Z of $C_{\mathcal{D}} \cup \mathit{fixcons}(X, \mathit{Occ}) \cup \mathit{fixcons}(Y, \mathit{Open}))$ such that $\alpha(n) \notin Z$ (Lemma 7)
iff the second lemma statement holds (Lemma 6). □

Lemma 9
Let $\mathcal{D}$ be a planning description that represents a conformant planning problem $\mathcal{PP}$ such that $\mathbf{I}(\mathcal{D})$ is in *GDT* form, and let n be a positive integer. Then, the following two statements are equivalent:

- there is some $X \subseteq \mathit{Occ}$ such that, for all $Y \subseteq \mathit{Open}$, there is some stable model of $C_{\mathcal{D}} \cup \mathit{fixcons}(X, \mathit{Occ}) \cup \mathit{fixcons}(Y, \mathit{Open})$; and
- there is some $X \subseteq \mathit{Occ}$ of the form $\{occ(a_1, 1), \ldots, occ(a_n, n)\}$ such that, for all $Y \subseteq \mathit{Open}$, there is some stable model of $C_{\mathcal{D}} \cup \mathit{fixcons}(X, \mathit{Occ}) \cup \mathit{fixcons}(Y, \mathit{Open})$.

Proof
The second statement trivially implies the first. In the other direction, assume that the first statement holds, we show that in this case the second statement also holds. Pick any $Y \subseteq \mathit{Open}$, and let M be some stable model of $C_{\mathcal{D}} \cup \mathit{fixcons}(X, \mathit{Occ}) \cup \mathit{fixcons}(Y, \mathit{Open})$. Given that the rule *Generate* belongs to $C_{\mathcal{D}}$, it follows that $M \cap \mathit{Occ}$ has the form $\{occ(a_1, 1), \ldots, occ(a_n, n)\}$. On the other hand, since M satisfies $\mathit{fixcons}(X, \mathit{Occ})$, it follows that $M \cap \mathit{Occ} = X$. Hence, X has the form $\{occ(a_1, 1), \ldots, occ(a_n, n)\}$, and the second statement holds. □

Proof of Theorem 6.1
Since $\mathcal{D}$ represents $\mathcal{PP}$, it follows that $\mathcal{PP} = \langle \Phi_{\mathbf{D}(\mathcal{D})}, SM(\mathbf{I}(\mathcal{D})), G \rangle$ where G is $SM(\mathbf{G}(\mathcal{D}) \cup \{\{f\} \leftarrow\mid f \in F\})$. Then, the program

$$\exists \mathit{Occ} \forall \mathit{Open}\ C_{\mathcal{D}}$$

is satisfiable
iff there is some $X \subseteq \mathit{Occ}$ such that, for all $Y \subseteq \mathit{Open}$, there is some stable model of

$C_{\mathcal{D}} \cup \mathit{fixcons}(X, Occ) \cup \mathit{fixcons}(Y, Open)$
iff there is some $X \subseteq Occ$ of the form $\{occ(a_1, 1), \dots, occ(a_n, n)\}$ such that, for all $Y \subseteq Open$, there is some stable model of $C_{\mathcal{D}} \cup \mathit{fixcons}(X, Occ) \cup \mathit{fixcons}(Y, Open)$ (Lemma 9)
iff there is some $X \subseteq Occ$ of the form $\{occ(a_1, 1), \dots, occ(a_n, n)\}$ such that every $s_0 \in SM(\mathbf{I}(\mathcal{D}))$ satisfies $\Phi_{\mathbf{D}(\mathcal{D})}(\{s_0\}, p) \subseteq G$ with $p = [a_0; \dots ; a_n]$ (Lemma 8)
iff there is some plan p of length n s.t. every $s_0 \in SM(\mathbf{I}(\mathcal{D}))$ satisfies $\Phi_{\mathbf{D}(\mathcal{D})}(\{s_0\}, p) \subseteq G$
iff there is some plan p of length n such that $\Phi_{\mathbf{D}(\mathcal{D})}(SM(\mathbf{I}(\mathcal{D})), p) \subseteq G$
iff there is some plan of length n that solves problem $\mathcal{PP}$. $\square$