Translation Algorithm for Negative Literals in Conformant Planning

Weisheng Li, Jiao Du
College of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065, China
Email: liws@cqupt.edu.cn, Dujiao19880429@126.com

Lifang Zhou
College of Software, Chongqing University of Posts and Telecommunications, Chongqing 400065, China
Email: Zhoulifang160966@sina.com

***Abstract*—The encoded negative literals in a conformant planning task will result in increasing state spaces. Getting a compact representation of state spaces is one of the most important issues in conformant planning. In this paper, a translation algorithm for negative literals is proposed to reduce the state spaces in a conformant planning task. The relationship between encoded literals is analyzed in detail. Based on the one-of relaxation technique in domain language, the algorithm is used to express the uncertain initial states and action effects in conformant planning. It converts formula one-of into a set of mutually exclusive literals with the relationship of mutual. The experiment study shows the efficiency of the proposed algorithm in pruning the state space in conformant planning tasks.**

***Index Terms*—negative literals, domain language, conformant planning, formula one-of, mutually exclusive literals**

I. INTRODUCTION

Conformant planning refers to solve the planning problems with incomplete information in an initial state and in state transitions. It decides whether there exists a linear sequence of actions for achieving the goal states from any initial belief state and resolution of the non-determinism in a planning problem [1]. Considering all possible initial states and transitions, the problem of conformant planning is more complex than that of classical planning as even under polynomial restrictions on plan length and the verification of plans is intractable in the worst case. Meanwhile, compared with the classical planning, a conformant planning task with uncertainty in initial states and action effects has greater state spaces [2].

The standard language for conformant planning is PPDDL (Probabilistic Planning Domain Definition Language) [3]. PPDDL is an extension to the Planning Domain Definition Language (PDDL) [4] and a standard for expressing the non-deterministic planning problems. The formal definition of the PPDDL is extended with an additional non-deterministic one-of statement. Because the one-of statement is defined for representing uncertain initial states and action effects, the number of literals in conformant planning is larger than that of the classical planning. It can also result in increasing the state spaces for conformant planning.

Getting a compact representation of state spaces is one of the most important issues in a conformant planning task. To decrease the state spaces of conformant planning, many conformant planners try to reduce the sum of encoded literals. Literals can be classified into positive and negative literals. Classical planning is based upon the closed world assumption: the atoms which do not exist in the states are false. The method of translating positive literals is as the same as that in classical planning. However, the translation of negative literals is a novel work. There is a Multi-valued Conformant Planning Tasks (MCPT) [5] algorithm which extends the Multi-valued Planning Tasks (MPT) [6] transform algorithm. In MCPT, when translating negative literals in conditions as $\neg p$, if the condition contains also some positive literals represented by the same variable v as p, there is no necessary to encode them at all. Otherwise a new derived variable not-p is introduced, and an axiom for each valued $d \in D_v \backslash \{p\}$ is generated. This is reasonable in the classical planning. However, in conformant planning the atoms that do not exist in the state are false or unknown. We improved the algorithm in the MCPT and noted as CPT-FDR [7]. Based our earlier work, a novel algorithm for translating negative literals is proposed to reduce the state spaces in a conformant planning task in this paper. The algorithm deals with the formula one-of in domain language to express the uncertain initial states and action effects in conformant planning. The algorithm converts formula one-of into a set of mutually exclusive literals with the relationship of mutual. The experiment study shows that the proposed algorithm is capable to translate most of the negative literals in a conformant planning task.

II. CONFORMANT PLANNING

Corresponding author: Weisheng Li.

doi:10.4304/jsw.8.1.39-46

Conformant planning involves generating plans under the condition that the initial states and action effects are nondeterministic and sensing is unavailable during plan execution.

A. Development

In 1998, Smith and Weld [8] gave a definition of conformant planning as in the case of all possible world states to develop valid plans without sensory information. In 2004, Brafman and Hoffmann [9] proposed that a task of uncertain initial states and action effects to generate plans without any sense capabilities during plan execution is conformant planning. The valid plan should be successful regardless of which particular initial world starting from and which action effects occur while executing an action. In 2006, conformant planning [10] was joined in IPPC (International Probabilistic Planning Competition) as conformant track for the first time. In 2008, conformant planning tracks [11] were partitioned to the part of uncertainty in IPPC secondly joined in this international competition. Conformant track is renamed as NOND (non-observable non-deterministic) track. NOND task is one without sensory information from agents and without certain initial belief states and action effects. In 2011, conformant track [12] continued to be a part of IPPC. In this time, it has been renamed as POMDPs (Partially-observable Markov Decision Processes) track and NOMDPs (No-Observation Markov Decision Processes). A conformant planner [13] can handle with NOMDPs and POMDPs problems. A conformant plan is a sequence of actions which will lead to the goal states with at least some probability p, which is predetermined before the planning task. Conformant planning problem [14] is the path-finding problems over a directed graph $G = <V, E>$ where V is the nodes in the graph and $E = <v_1, v_2>$ is the directed edges in the graph. The nodes are the sets of all belief states, which express the states of the world that are assumed possible to the agent. The edges illustrate that executing action has changed the current state v_1 to the next state v_2.

The first conformant planner CGP (Conformant Graphplan) was based on the classical planner Graphplan. This planner was designed to deal with the problem with uncertainty in the initial states and certainty in action effects. The valid plan of CGP is distributed into two phases: build planning graph and extract the valid plans. Then Brafman and Hoffmann designed CFF (Conformant Fast Forward) planner to handle the initial states with uncertainty. CFF is a domain independent planning system. It extends the classical planner FF (Fast Forward) which handles uncertain planning task expressed in the form of a CNF formula. It defines known propositions as the proposition is always true in all possible belief states and replaces CNF with 2-CNF projection of the formula that captures the true belief state semantics. Next a large number of conformant planners are springing up since 2006 [10]. Most of conformant planners are to translate the conformant problem into classical problem, such as CGP, CFF, CMBP and t0.

In Probabilistic-FF conformant planner, the conformant problem is that the initial belief state with uncertainty and uncertain action effects. Especially it used the Bayesian networks to present all belief states in the planning task. In t0 planner [15], it used some algorithm in classical planning for solving conformant problems with incomplete information. Finding sequence of actions between belief states is the vital part in planning under no observations. Planner t0 use propositional logic, like in SATPLAN. The search is to construct a CNF with all possible plans. It introduced the function $K(p)$, meaning that translating from conformant problem p into classical problem $K(p)$. All the literals in the conformant problem are changed into conjunctive formula of KL and $\neg KL$. KL is known literal L while $\neg KL$ is unknown literal L.

B. Conformant Planning Tasks in PPDDL

In [7], an approach to translating the PPDDL-based conformant planning tasks into FDR (Finite Domain Representation) state variable is described. The approach extends the FDR algorithm to settle uncertain initial conditions and the non-deterministic operator effects, is introduced to reduce the size of belief states.

A PPDDL task for conformant planning can be described by a 4-tuple:

$$\prod = <I_0, I_G, A, O> \tag{1}$$

The components in (1) are described as follows.

1) I_0 is the set of all initial belief states. It is composed of conjunction and disjunction of ground atoms over the object named the initial belief state. The conjunction formula represents for the sets of atoms in the same belief states, while the disjunction formula is the relationship between different possible belief states.

As the initial belief state is uncertain, some new operators, including or and one-of operators, are defined as follows.

Operator or $(l_1, l_2, \ldots, l_n)$ is defined in the PPDDL and it is equal to the sets $\{l_1, l_2, \ldots, l_n\}$. One has

$$\{l_1, l_2, \ldots, l_n\} = l_1 \bigcup l_2 \bigcup \ldots \bigcup l_n \tag{2}$$

Operator one-of $(l_1, l_2, \ldots, l_n)$ is defined in PPDDL and it is used to represent for the uncertain initial states. The one-of statement has the form:

$$\text{one-of } (e_1, e_2, \ldots, e_n) \tag{3}$$

where e_k (k=1, 2, …, n) are the PPDDL effects.

2) I_G is the closed formula called the goal formula. It is the conjunction formula of atoms or negative atoms.

3) A is a finite stratified set of the PPDDL axioms. A PPDDL axiom is a pair $\langle \varphi, \psi \rangle$ such that φ is a first-order atom and ψ is a first order formula with free (ψ) $\subseteq$ free (φ). The axiom formula $\langle \varphi, \psi \rangle$ is the same as $\varphi \leftarrow \psi$, where φ is the head of this formula and ψ is the body of this formula.

The set A of PPDDL axioms is stratifiable only if there is a total preorder $\prec$ on the predicate symbols of A such that for each axiom where predicate Q occurs in the head, having $P \prec Q$ for all predicates P occurring in the body,

and $P \prec Q$ for all predicates P occurring in a negative in the transition of the body to negation normal form.

Stratifiability of a set of axiom formulas is with the purpose for making sure that the outcome of axiom evolution is well-defined. Suppose that if the axiom formula is not stratifiable, it would be possible to specify rules of the form as $P(x)$ is true if $P(x)$ is false. $P \prec Q$ means that the truth value of atoms over Q must be determined after the truth value of atoms over P.

4) O is a finite set of operators. A schematic operator o_1 is showed as $\langle \chi, e \rangle$, where χ the first-order formula is defined as precondition of the operator o_1 and e is the conjunction or disjunction sets of first-order formula defined as effect of the operator o_1. Especially, the PPDDL effects of operator are recursively defined by finite application of the following rules:

a) A literal l is a first-order formula. The effect is expressed by l is called a simple effect.
b) If $e_1 \ldots e_n$ are the effects of one operator, then $e_1 \wedge e_2 \wedge \ldots \wedge e_n$ is called a conjunctive effect.
c) $\chi \rhd e$ is called a conditional effect if χ is a first-order formula and e is the effect of one operator.
d) If $v_1, v_2, \ldots, v_n$ are the variables in the first-order formula and e is an effect of one operator, then $\forall v_1, v_2, \ldots, v_n: e$ is called a universally quantified effect or a universal effect.
e) If $e_1 \ldots e_n$ with e_i $(1 \le i \le n)$ is simple effect as described in the first rule are the effects in the formula one-of $(e_1, e_2, \ldots, e_n)$, then every effect e_i $(1 \le i \le n)$ is in one-of formula is called a non-deterministic effect.

Free variable is the parameter of one operator. The free variables of simple effects are defined as for literals in first-order logic. Free variables of other effects are defined with following rules which are shown in (4), (5) and (6).

$$\text{free}(e_1, e_2 \ldots e_n) = \text{free}(e_1) \cup \text{free}(e_2) \cup \ldots \cup \text{free}(e_n) \quad (4)$$

$$\text{free}(\chi \rhd e) = \text{free}(\chi) \cup \text{free}(e) \quad (5)$$

$$\text{free}(\forall v_1, v_2, \ldots, v_n: e) = \text{free}(e) \setminus \{ v_1, v_2, \ldots, v_n \} \quad (6)$$

In the definition of the PPDDL task for conformant planning, the operator O defines the translation of the world state. If the current state satisfies the precondition of one operator, then the operator may be executed and meanwhile it has renamed as action. The executing action leads to a new state which is like the old one except that it is modified in certain ways specified by the effect of the action. An operator with parameters cannot be applied directly. It must be grounded by substituting concrete objects for the parameters.

C. Conformant Planning Tasks in FDR

Conformant planning task in a finite-domain representation is given by a 5-tuple:

$$\prod = \langle V, I_0, I_G, A, O \rangle \quad (7)$$

The components in (7) are described as follows.

1) V is a finite set of FDR state variables. Every variable $v \in V$ owns an associated finite domain represented by sets of values, D_V. The state variable is portioned into two kinds: fluent and derived variable. The fluent is affected by operators or occurring in I_0. The value of the fluent is changed from $value_1$ to $value_2$. The derived variable is computed by evaluating axioms. The domain of derived variables must contain the undefined value $\perp$. The undefined value means the value of this variable is unknown or does not matter. As the planning problem is the reasoning problem of a totally or partially ordered set of operators that achieves a specified goal from a given initial state. So a total world state assigns defined values to all state variables, while a partial world state is to assign the undefined value. The undefined value is for representing the partial world state.

2) I_0 is a set of completely specified all possible initial world, called the initial belief state. Every initial world is consisted of the conjunctions of state variables assignment over V.

3) I_G is a finite set of DNF formula, called the goal of the task. Each goal is composed of the conjunction of state variables assignment over V.

4) A is a finite set of axioms over V. It is explained as a triple $\langle cond, v, d \rangle$, where $cond$ is the condition or the body of the axiom, and pair $\langle v, d \rangle$ is the head of the axiom. In the pair $\langle v, d \rangle$, v is a derived variable named an affected variable, and $d \in D_v$ is the new value for V.

5) O is a set of operators with the form $\langle pre, post, prv \rangle$, where it is denoting the pre-, post and prevail-condition respectively. In the form $\langle pre, post, prv \rangle$, pre and prv is the precondition of the operator and $post$ is the effect of the operator. The variable in pre is affected variable while the variable in prv is unaffected variable. The effect $post$ is defined as a tripe $\langle cond, v, d \rangle$, where $cond$ is a partial variable assignment, v is a fluent affected by the operator and d is a new value for V. For every operator $o = \langle pre, post, prv \rangle \in O$, it must satisfy two restrictions shown as follows.

For all $v \in V$, if $pre[v] \neq \perp$, then

$$pre[v] \neq post[v] \neq \perp \quad (8)$$

For all $v \in V$,

$$post[v] = \perp \quad (9)$$

or

$$prv[v] = \perp \quad (10)$$

Thirdly, the state space of the conformant planning task in a finite-domain representation $\prod = \langle V, I_0, I_G, A, O \rangle$ is denoted as $P(\prod)$. The formula $P(\prod)$is a directed graph:

$$G = \langle V, E \rangle \quad (11)$$

where V is the sets of state variable V. $E = \langle v_1, v_2 \rangle$ if exists one operator with the formula $\langle pre, post, prv \rangle$ or

axiom with the formula $\langle cond, v, d \rangle$, such that $pre \subseteq s$ (a state of conformant planning task) and $s'(v) = d$ for all effects or axiom $\langle cond, v, d \rangle$ such that $cond \subseteq s$ and $s'(v) = s(v)$ for all other fluent.

Finally, the conformant planning task is the problem of planning with the following definition.

Given a conformant planning task by 5-tuple: $\prod = <V, I_0, I_G, A, O>$ with initial belief state I_0 and goal set I_G, the planning computes some paths in directed graph of $P(\prod)$, or proves that I_G cannot be achieved in the task. The paths which all encode the same action sequence, should reach one of goals in I_G respectively, regardless of which initial world the planning start from and which action effects occur.

In [7], the difference among FDR, MCPT and CPT-FDR in a finite-domain representation is summarized, which is shown in Table I.

TABLE I.
THE DIFFERENCE AMONG FDR, MCPT AND CPT-FDR

Property	FDR	MCPT	CPT-FDR
Fluents in state variables	affected by operators	affected by operators	affected by operators or occurring in initial belief state
Initial state	complete information	uncertain	uncertain
Operator effects	deterministic	deterministic	non-deterministic
others	extended states	-	extended belief states

In Table I, "-" means nothing.

III. LITERALS AND ONE-OF FORMULA

Literals in planning task are the atoms or negate atoms for representing the propositions in the real world. The atom in literals is well formed formula including antecedent and consequent. Literals can be classified into two kinds. First one is the positive literal a is of the form P (t_1, t_2, …, t_n), while another is the negative literal $\neg a$ is of the form $\neg P$ (t_1, t_2, …, t_n) where P is a predicate and the terms t_i in formula are constants or variables.

The number of literals is a key part consisting of state space for the conformant planning task. Translating positive literals of the form P (t_1, t_2, …, t_n) in classical planning is to encode into Boolean variables and numeric variables. While translating negative literals of the form $\neg P$ (t_1, t_2, …, t_n) is a hard work. As whether the atom is true or not in the states of conformant planning, known true atoms and known false atoms are important for computing conformant plan. However, classical planning is based on closed world assumption that the atoms which do not exist in the current state are always false. Thus it is unnecessary to encode the negative literals directly in classical planning. Conformant planning is the task with uncertain initial states and action effects and large number of negative literals exists in the conformant planning. In the first conformant planner CGP which is dealing with the problem with uncertainty in the initial states and certainty in action effects. The valid plan of CGP is parted into two phases: build planning graph and extract the valid plans. Firstly, to build planning graphs for all possible world states and then prune planning graphs into only a planning graph by connecting the nodes in graphs. Secondly, if all goal states in the sets of currently nodes it can generate the valid plan. The negative literals in CGP are converted into the node of the planning graph for representing all possible world states. In 2006, conformant planner CFF transforms the conformant planning into a search problem in the state space of all possible belief states and distinguishes between the belief states and world states. In this conformant planning system, it defines known propositions as the proposition is always true in all possible belief states. By defining the known literals, the atom in the process of find solution is known true or false. A known literal a is true, so the literal a is true in all states for computing valid plan for conformant planning.

The standard definition language of conformant planning is PPDDL (Probabilistic Planning Domain Definition Language) added with the one-of formula proposed in the first conformant track completion, 5th International Planning Competition: Non-deterministic Track. In IPPC2008 the standard definition language is unchanged. However, RDDL (Relational Dynamic Influence Diagram Language) [17], a new language, is proposed by Sanner. As PDDL or PPDDL cannot represent some domain models such as CTM (cell transition model), RDDL is coming up. RDDL is integrated by PDDL family, PPDDL, stochastic programs, influence diagrams, etc. Initial belief states, actions, goal states are parameterized variables and the evolution of the planning problem is specified via functions over the next state variable that can be obtained in the case of a particular problem instance defining possible domain objects. In sum the new language is just a factored MDP (Markov Decision Processes) or POMDP. However most of domains can be expressed by PPDDL + one-of, defined in IPPC2006.

PPDDL is an extension to PDDL (Planning Domain Definition Language) for expressing planning domains with probabilistic effects. PPDDL1.0 [3] is the input language for the probabilistic track of the 4th International Planning Competition. The semantics of PPDDL1.0 is (probabilistic $p_1e_1 \ldots p_ke_k$) where effect e_i occurs with probability p_i ($p_i \geq 0$ && $\sum_{i=1\text{-}n} p_i = 1$). For example, the effect (probabilistic 0.05 (toilet-clogged)) means that with probability 0.05 the state variable toilet-clogged turns into true after executing this probabilistic effect, whereas with probability 0.95 the state is unchanged in the next state. To express the conformant planning the IPC puts forward new standard domain language. The new language is PPDDL+ one-of formula. It is extended the PPDDL1.0 with an additional non-deterministic statement, the counterpart of the probabilistic statement for non-deterministic models. The origin form of PPDDL1.0 is (probabilistic p_1e_1, …, p_ke_k) while the changed form is one-of (l_1, …, l_n) where l_i ($1 \leq i \leq n$) is initial belief state or action effect. On account of translating the form, many negative literals exist in conformant planning task. The one-of formula can be expressed as one-of (s_1, s_2, …, s_n) where s_i ($1 \leq i \leq n$) is the

initial belief state especially n is the total number of initial belief states or one-of $(a_1, a_2, \ldots, a_n)$ where a_i $(1 \le i \le n)$ is one of action effects especially n is the total number of action effects. The former is standing for uncertain initial states and the latter is for uncertain action effects.

IV. Translation Algorithm

The one-of formula leads in many literals because of one-of $(l_1, l_2 \ldots l_n) \Rightarrow l_i, \sum_{j=1\sim n} \neg l_j$ $(j \neq i)$. When executing such initial state or effect, one of the l_i is chosen and applied to the current state. The numbers of negative literals are augments with the number of the one-of formula. Supposing that there is a formula stating as one-of $(l_1, l_2, \ldots, l_n)$ and the number of negative literals is n-1. The count of negative literals is computed $n \times (D_n\text{-}1)$ where n is the number of one-of formulas and D_n is sum of literals in one-of formula. Converting negative literals is complex. In CFF [9], it adopts a new derived variable not-p with domain values $\{\Delta, \perp\}$ and generates an axiom $(v = d) \rightarrow$ (not-p: $= \Delta$) for each value $d \in D_v \backslash (p)$. not-$p = \Delta$ can serve as a translation of the literal $\neg p$. The new derived variable not-p is introduced for representing the negative literal in the planning task. For this reason the state space of planning task with n negative literals is increased by 2^n. Negative literals appear in initial states, action precondition, action effects and goal states. And these literals are mostly in initial states, action effects and goal states. Negative literals in initial states can be classified into real negative literals and negative literals implied by the positive literals. Such as Initial states= {ontable (a), handempty (), clear (a), ontable (a), clear (b), $\neg$ on (b, a), $\neg$ holding (b), $\neg$ on (a, b)}, a variable with domain values represents set {clear (a), holding (a)} which is mutually exclusive. And holding (a), $\neg$ holding (a) are mutually exclusive, positive literal clear (a) impliedly present negative literal $\neg$ holding (a). The rest literals are real negative literals. Negative literals in goal states are also two kinds: impliedly presented by positive literals and need not encode the negative literals. Especially the second kind means the planning task with the negative literals is not solvable. The negative literals mostly in these two parts as the two forms of one-of are defined for uncertain initial states and action effects. Experiments with the domains in IPPC2006 [10], IPPC2008 [11] show that the negative literals are in the initial states except for the btuc and bmtuc. The negative literals in btuc and bmtuc are from the uncertainty in initial states and action effects and the negative literals from action effects is only one. In this paper, the algorithm only handles with the negative literals in initial states. Formula one-of$_1$ $(l_{11}, l_{22}, \ldots, l_{1n})$, one-of$_2$ $(l_{21}, l_{22}, \ldots, l_{2n})$, described in section II means that $\{l_{11}, l_{22}, \ldots, l_{1n}\}$ or $\{l_{21}, l_{22}, \ldots, l_{2n}\}$ are mutually exclusive while $\forall\, l_{1i}, l_{2j}$ $(l_{1i} \in$ one-of$_1$, $l_{2j} \in$ one-of$_2$) && $(1 \le i, j \le n)$ are mutually compatible. Based on the relationship between literals in initial states, the algorithm for translating negative literals is explained as follows.

Algorithm 1. Algorithm for translating negative literals:

Input: sets of formula one-of F, sets of invariant I
Output: sets of state variables S
$F =$ (one-of$_1 \ldots$one-of$_n$)
$S_i = \{\}$
$i = 0$
Repeat
$i = i + 1$
$F = \{l_{i1} \ldots l_{in} \mid$ one-of$_i$ $(l_{i1} \ldots l_{in})\}$
Build new variable $v = \{0: l_{i1}, 1: l_{i2} \ldots n\text{-}1: l_{in}\}$
For each *invariant* in I
$v = v - (v \cap invariant)$
if v$\neq \varnothing$
Insert v in S_i
End of for
Until $S_i == S_{i-1}$
Return S_i

Algorithm 1 is divided into three steps. Firstly, find the entire formula one-of in the input file problem. As most of the one-of statements exist in the initial states, we just change all the formula one-of $(s_1, s_2, \ldots, s_n)$ into sets of $\{s_1, s_2, \ldots, s_n\}$ where s_i $(1 \le i \le n)$is the literal represents one of the uncertain initial belief states. Secondly, every set of literals in each formula one-of is encoded as a state variable. Such as one-of (s_1, s_2, s_3, s_4) is encoded into a state variable var0 (0: s_1, 1: s_2, 2: s_3, 3: s_4). Supposing the initial states holds m formulas one-of, the total number of new state variables is m. The number of new state variables equals to the number of formulas one-of. The state space of conformant planning is $n \times D_n$, where n is the number of state variables and D_n is the range of state variable n's domain values. To reduce the state space of this planning, decrease the factor n can be a valid method. The second step has created large number of state variables. With the purpose of getting rid of these new variables, the third step of the algorithm is to simplify the state variable. The process of simplify is to delete the domain value which is the interests of the new state variable of the formula one-of and the invariant state variable of the whole action effects. If the literals in new state variable are all in the invariant state variables, delete this new state variable. Otherwise retain the literals in new state variable.

V. Experimental Study

To prove the efficiency of the proposed algorithm in this paper, we list some experimental results with some problems of standard domain in IPPC2006.

The red line labeled with unchanged shows the number of encoded negative literals without the algorithm while the green line labeled with changed shows the number of encoded negative literals with the algorithm in Fig. 1, Fig. 2, Fig. 3 and Fig. 4, respectively.

Fig. 1 shows the result of the number of encoded negative literals with 3 problems from blocksworld domain in IPPC2006. The algorithm, described in Figure1, reduces the number of encoded literals from 3, 5 and 6 to 0, 0, and 0. The algorithm translates the negative literals in formula one-of efficiently. Problems in blocks

world are the task with uncertain initial belief states and certain action effects. Formula one-of exists in initial states.

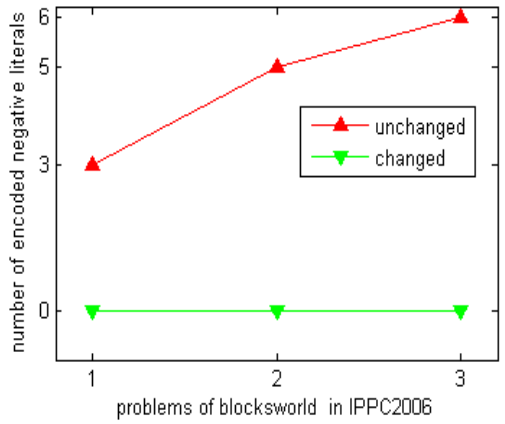

Figure 1. The number of encoded negative literals in blocksworld problem.

Fig. 2 shows that the result of number of encoded negative literals in uts problem in IPPC2006.

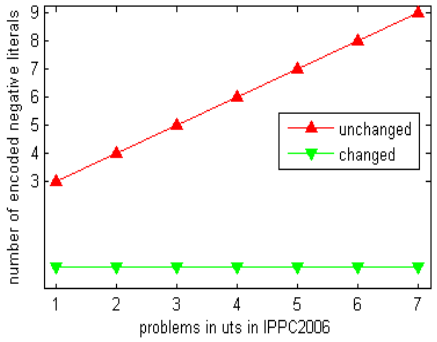

Figure 2. The number of encoded negative literals in uts.

7 problems in uts problem are chosen in the experiments and the algorithm changed the number from 3, 4, 5, 6, 7, 8 and 9 to 0, 0, 0, 0, 0, 0 and 0, respectively. These problems are as same as the problems in blocksworld problem and they are the tasks with only uncertainty in initial states.

Take one problem in uts in IPPC2006 to explain the algorithm in detail as follows.

Problem: uts_cycle_3
(one-of (and (at_node n_0) (visited n_0))
(and (at_node n_1) (visited n_1))
(and (at_node n_2) (visited n_2))),
(one-of (and (edge_label n_0 n_1 l_1) (edge_label n_0 n_2 l_2))
(and (edge_label n_0 n_1 l_2) (edge_label n_0 n_2 l_1))),
(one-of (and (edge_label n_1 n_2 l_1) (edge_label n_1 n_0 l_2))
(and (edge_label n_1 n_2 l_2) (edge_label n_1 n_0 l_1))),
(one-of (and (edge_label n_2 n_0 l_1) (edge_label n_2 n_1 l_2))
(and (edge_label n_2 n_0 l_2) (edge_label n_2 n_1 l_1))).
1)f = {{ Atom at_node (n_0), Atom at_node (n_1), Atom at_node (n_2)},
{Atom edge_label (n_0 n_1 l_1), Atom edge_label (n_0 n_2 l_2),
Atom edge_label (n_0 n_1 l_2), Atom edge_label (n_0 n_2 l_1),
{Atom edge_label (n_1 n_2 l_1), Atom edge_label (n_1 n_0 l_2),
Atom edge_label (n_1 n_2 l_2), Atom edge_label (n_1 n_0 l_1),
{Atom edge_label (n_2 n_0 l_1), Atom edge_label (n_2 n_1 l_2),
Atom edge_label (n_2 n_0 l_2), Atom edge_label (n_2 n_1 l_1).
2)I = {{Atom at_node (n_0), Atom at_node (n_1), Atom at_node (n_2)},
{Atom edge_label (n_0 n_1 l_1), Atom edge_label (n_0 n_2 l_2),
{Atom edge_label (n_0 n_1 l_2), Atom edge_label (n_0 n_2 l_1),
{Atom edge_label (n_1 n_2 l_1), Atom edge_label (n_1 n_0 l_2),
{Atom edge_label (n_1 n_2 l_2), Atom edge_label (n_1 n_0 l_1),
{Atom edge_label (n_2 n_0 l_1), Atom edge_label (n_2 n_1 l_2),
{Atom edge_label (n_2 n_0 l_2), Atom edge_label (n_2 n_1 l_1).
3)S = ∅
state variable = S + I

The Problem uts_cycle_3 explains one problem of domain uts in Fig.1. The initial states hold 18 negative literals and by the algorithm the number of encode negative literals has changed into 0. The algorithm is parted into three and using 1), 2) and (3) to represents these three steps.

However, some problems shown in Fig. 3 hold no negative literals. And the algorithm does not translate any negative literals at all. These 7 problems are from domain coins. The reason for this is that no formula one-of exists in the input file.

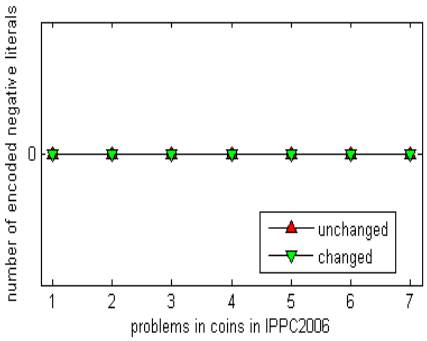

Figure 3. The number of encoded negative literals in coins.

Finally, the problems in Fig. 4 indicate that the number of encoded negative literals has converted from 2, 3, 4, 5 and 6 to 2, 3, 4, 5 and 6. The algorithm does not convert any negative literals. Why.

Take one of problems in comm to resolve this question.

Problem: comm_5_2
one-of (noisy p_0, not (noisy p_0))
one-of (noisy p_1 not (noisy p_1)
1) f = {{ Atom noisy (p_0), NegatedAtom noisy (p_0)},
{Atom noisy (p_1), NegatedAtom noisy (p_1)}.
2) I = {{ Atom current-stage (s_0), Atom current-stage (s_1),

Atom current-stage (s_2), Atom current-stage (s_3), Atom current-stage (s_4)}, {Atom read (p_0), Atom in-channel (p_0)}, {Atom read (p_1), Atom in-channel (p_1}.

3) S = {{Atom noisy (p_0), NegatedAtom noisy (p_0)}, {Atom noisy (p_1), NegatedAtom noisy (p_1)}.

state variable = $S + I$

Based on the above comm._5_2 problem, it is shown that the negative literals existing in literals of one-of statements are real negative literals. Thus, they are needed to be encoded.

VI. Conclusions

A new translation algorithm for negative literals in conformant planning is introduced in the paper. The paper firstly gives an overview of the development of conformant planning. The definition of conformant planning is that of deciding whether there exists a linear sequence of actions for achieving the goal states from any initial belief state and resolution of the non-determinism in the problem. Then it lists some planners for solving problems in conformant planning tasks. Secondly, it defines conformant planning tasks from four aspects: PPDDL tasks, conformant tasks in FDR, the state space of the conformant tasks in FDR and the process of planning for conformant tasks in FDR. The conformant planning tasks is the basis of the algorithm as described in part: translation algorithm. Next, the literals in the conformant planning tasks and one-of formula existing in the standard domain language PPDDL are explained. Based on the analysis of the relationship between literals defined with PPDDL, the literals are divided into some kinds by the relationship between them. A translation algorithm is proposed in view of the theories stating in the front of this part. The translation algorithm is consisting of three steps. This algorithm builds a connection between literals in one-of formula and invariants computing by the algorithm proposed by Helmert. It not only considers every literal in one-of formula but also the literal in invariants with the purpose of finding mutually exclusive literals. There exists intersection between the literals in one-of formula and invariants. By using this intersection, it can ensure that the state variable computing by the algorithm is the finite state variable representing maximum literals. It can reduce the state space of the conformant planning tasks.

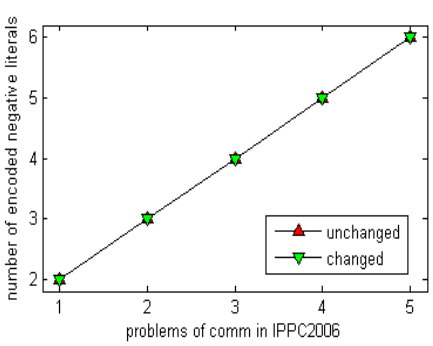

Figure 4. The number of encoded negative literals in comm problem.

The translation algorithm in this paper was proposed on the basis of PPDDL which is the standard domain definition languages of conformant planning added with one-of relaxation technique. . Two forms of formula one-of are one-of (s_1, s_2, …, s_n) and one-of (a_1, a_2, …a_n). The former is for the uncertain initial belief states however the latter is for the uncertain action effects. The algorithm is to translate the negative literals in the former form. Experiments with the 22 problems of conformant track in IPPC2006 show that the algorithm can translate most of the negative literals in initial world states. But conformant planning is the task with uncertainty in initial states and action effects. The algorithm in this paper can only handle with the uncertainty in initial states. To analyze the form of one-of on the latter, negative literals also exist in action effects. In order to translate all the negative literals, one can combine the uncertain action effects with the invariant synthesis inferred by Helmert.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China (No. 61142011, No. 61100114), and the Key Project of Chinese Ministry of Education (No. 210184).

References

[1] H. Palacios and H. Geffner. “From conformant into Classical Planning: Efficient Translations That May be Complete Too”, *Proc. ICAPS-07*, 2007.

[2] B. Bonet and R. Givan, 5th International Planning Competition: Non-deterministic Track Call for Participation, 2005.

[3] H. L. S. Younes, M. L. Littman, “PPDDL1.0: An Extension to PDDL for Expressing Planning Domains with Probabilistic Effects”, Technical Report CMU-CS-04-167, *Carnegie Mellon University*, Pittsburgh, Pennsylvania, USA, 2004.

[4] A. Gerevini, D. Long, “Plan Constraints and Preferences in PDDL3”, Technical report, R.T. 2005-08-07, *University of Brescia*, Brescia, Italy, 2005.

[5] J. Zhao, J. Sun, M. Yin, “Translating PDDL tasks into multi-valued conformant planning tasks”, *Proc. the 4th Int. Conf. Fuzzy Systems and Knowledge Discovery*, *IEEE Computer Society*, Washington, DC, USA, pp. 204-208, 2007.

[6] M. Helmert, “The Fast downward planning system”, *Journal of Artificial Intelligence Research* (*JAIR*), Vol.26, pp.191-246, 2006.

[7] W. S. Li, Z. Zhang and W. X. Wang, “CPT-FDR: An Approach to Translating PPDDL Conformant Planning Tasks into Finite-Domain Representations”, Chinese Journal of Electronics, vol. 21, pp. 53-58, 2012.

[8] D. E. Smith and D. S. Weld. “Conformant Graphplan”, *Proc. AAAI’98*, 1998.

[9] R. I. Brafman and J. Hoffmann, “Conformant Planning via Heuristic Forward Search: A New Approach”, *Artificial Intelligence*, vol. 170, pp. 507-541, 2006.

[10] B. Bonet and B. Givan, “Results of Conformant Track in the 5th International planning competition”, *Proc. IPC* 2006.

[11] D. Bryce and O. Buffet, “International Planning competition Uncertainty Part: Benchmarks and Results”, *Proc. IPC*, 2008.

[12] S. Sanner and S. Yoon., "IPPC Results Presentation", *Proc. ICAPS 2011*, pp. 16-17, 2011.
[13] A. Olsen. "Pond - Hindsight: Applying Hindsight Optimization to Partially - Observable Markov Decision Processes", *Utah State University*, January 2011.
[14] B. Bonet, H. Geffner, "Planning with incomplete information as heuristic search in belief space", *Proc.5th Int. Conf. on Artificial Intelligence Planning and Scheduling*, AAAI-Press, pp.52-61, 2000.
[15] Hector Palacios and Hector Geffner, From Conformant into Classical Planning: Efficient Transitions That May Be Complete Too. *Department de Technologia University Pompeu Fabra, ICAPS-2007*. 2007.
[16] M. Helmert, "Concise finite-domain representations for PDDL planning tasks", *Artificial Intelligence*, Vol.173, No.5-6, pp.503-535, 2009.
[17] S. Sanner, "Relational Dynamic Influence Diagram Language (RDDL): Language Description", *NICTA and the Australian National University*, pp. 1-4, 2011.

Weisheng Li was born in Anyue, Sichuan Province, China on Dec. 30, 1975. He graduated from School of Electronics & Mechanical Engineering at Xidian University in July 1997. He received M.S. degree and Ph.D. degree from School of Electronics & Mechanical Engineering and School of Computer Science & Technology at Xidian University in July 2000 and July 2004, respectively. Currently he is a professor of Chongqing University of Posts and Telecommunications. His research focuses on intelligent information processing and pattern recognition.

Jiao Du was born in Changshou, Chongqing City, China on Apr. 29, 1988. She graduated from Department of Computer Science and Technology at Jinggangshan University in July 2010. She received B. Admin. (Bachelor of Administration) from Jinggangshan University in July 2010. Currently she is a postgraduate of Chongqing University of Posts and Telecommunications. Her research focuses on intelligent planning.

Lifang Zhou was born in Tianshui, Gansu Province, China on Jul. 12, 1975. She received M.S. degree from School of Computer Science & Technology at Chongqing University of Posts and Telecommunications in July 2007. Currently she is a PhD. candidate of College of Computer at Chongqing University. Her research focuses on intelligent planning. Her research focuses on pattern recognition and image processing.