Advanced Petri Nets and the Fluent Calculus

Steffen Hölldobler and Ferdian Jovan

International Center for Computational Logic
Technische Universität Dresden, 01062 Dresden, Germany
sh@iccl.tu-dresden.de ferdian.jovan@gmail.com

Abstract. In this paper we discuss conjunctive planning problems in the context of the fluent calculus and Petri nets. We show that both formalisms are equivalent in solving these problems. Thereafter, we extend actions to contain preconditions as well as obstacles. This requires to extend the fluent calculus as well as Petri nets. Again, we show that both extended formalisms are equivalent.

1 Introduction

It is widely believed that humans generate models and reason with respect to these models [14]. It is less widely believed that logics can be used to adequately model human reasoning [3]. Based on ideas first presented in [18], Hölldobler and Kencana Ramli [10] have developed a logic based on weakly completed logic programs and interpreted under the three-valued Łukasiewicz semantics; this logic was shown to adequately model human reasoning scenarios like the suppression and the selection task by generating a least model of an appropriate logic program and reasoning with respect to this least model [6,7]. Moreover, it was shown that there is a connectionist realization of this approach based on the core method [11,1].

However, human reasoning is much more complex than the above mentioned scenarios and involves – among others – reasoning about actions and causality including compositionality, concurrency, quick reactions, and resilience in the face of unexpected events. An architecture for such actions was developed in [2] based on extended Petri nets. Unfortunately, there is a huge gap between Petri nets and the logic developed by Hölldobler and Kencana Ramli and it is not at all obvious how the two approaches can be combined. Moreover, a close inspection of [2] revealed that some concepts are only specified procedurally.

A central notion in Petri nets are tokens which are consumed and produced when executing an action. Likewise, in the equational logic programming approach to actions and causality presented in [12] resources are used. The approach was later called *fluent calculus* in [19]. The logic programs in the fluent calculus admit least models and reasoning is performed with respect to these models. Hence, the fluent calculus seems to be a promising candidate to add reasoning about actions and causality to the human reasoning approach of Hölldobler and Kencana Ramli.

The goal of the research presented in this paper is to understand the relation between the fluent calculus and the extended Petri networks used in [2]. To this end, we will rigorously define various classes of planning problems, we will map these problems into Petri nets and into the fluent calculus, and we formally prove that there is a one-to-one correspondence between the two approaches in solving such problems.

The paper is structured as follows: Following the introduction in Section 1 we will present main notions and notations in Section 2. Conjunctive and advanced planning problems are discussed in Sections 3 and 4. In the final Section 5 we will discuss our results and point to future work. Due to lack of space we cannot include proofs; they are worked out in detail in [15] if not stated otherwise.

2 Preliminaries

Multisets Multisets are generalizations of sets, where members may occur more than once. In this paper, multisets are depicted with the help of the parenthesis $\dot{\{}$ and $\dot{\}}$. $\dot{\emptyset}$ denotes the empty multiset and $\dot{\cup}$, $\dot{\cap}$, $\dot{\subseteq}$, $\dot{\setminus}$, $\dot{=}$, and $\dot{\neq}$ denote the usual operations and relations on multisets, viz. multiset-union, -intersection, -subset, -difference, -equality, -inequality, respectively. Moreover, $x \in_k \mathcal{M}$ holds if and if x occurs exactly k times in the multiset $\mathcal{M}$, where $k \in \mathbb{N}$.

Petri Nets A *Petri net* is a tuple $(\mathcal{P}, \mathcal{T}, \mathcal{F})$, where $\mathcal{P}$ and $\mathcal{T}$ are finite sets called *places* and *transitions*, respectively, $\mathcal{P} \cap \mathcal{T} = \emptyset$, and $\mathcal{F} \dot{\subseteq} (\mathcal{P} \times \mathcal{T}) \dot{\cup} (\mathcal{T} \times \mathcal{P})$. A *marking* is a finite multiset $\mathcal{M}$ over $\mathcal{P}$; its elements are called *tokens*. The *pre-set* $\bullet t$ of $t \in \mathcal{T}$ is a finite multiset with $p \in_k \bullet t$ iff $p \in \mathcal{P} \wedge (p,t) \in_k \mathcal{F}$. The *post-set* $t\bullet$ of $t \in \mathcal{T}$ is a finite multiset with $p \in_k t\bullet$ iff $p \in \mathcal{P} \wedge (t,p) \in_k \mathcal{F}$.

Let $\mathcal{N} = (\mathcal{P}, \mathcal{T}, \mathcal{F})$ be a Petri net and $\mathcal{M}$, $\mathcal{M}'$, and $\mathcal{M}''$ be markings. $t \in \mathcal{T}$ is *enabled* at $\mathcal{M}$ in $\mathcal{N}$ iff $\bullet t \dot{\subseteq} \mathcal{M}$; an enabled transition t can *fire* leading to $\mathcal{M}'$, denoted by $\mathcal{M} \xrightarrow{[t]} \mathcal{M}'$, where $\mathcal{M}' \doteq (\mathcal{M} \dot{\setminus} \bullet t) \dot{\cup} t\bullet$. *Firing sequences* are inductively defined as follows: $\mathcal{M} \xrightarrow{[]} \mathcal{M}$; if $\mathcal{M} \xrightarrow{[t]} \mathcal{M}'$ and $\mathcal{M}' \xrightarrow{w} \mathcal{M}''$ then $\mathcal{M} \xrightarrow{[t|w]} \mathcal{M}''$, where w is a list of transitions. A *firing sequence from $\mathcal{M}$ to $\mathcal{M}'$ of $\mathcal{N}$* is a firing sequence which starts from $\mathcal{M}$ and yields $\mathcal{M}'$.

Equational Logic Programming We assume the reader to be familiar with first-order predicate logic with equality and, in particular, with equational logic programming as, for example, presented in [9,17,13].

Fluents and Fluent Terms In planning, the notion of a fluent is often used to describe an item which may be present in one state but not in the next state. In the fluent calculus, *fluents* are non-variable terms built over some alphabet like a, $f(a)$, or $f(X)$, where a is a constant, f a function symbol, and X a variable; this alphabet must not contain the binary function symbol $\circ$ and the constant 1 as these symbols are used to represent multisets of fluents; *ground fluents* are fluents which do not contain an occurrence of a variable (e.g., a and

$f(a)$); *simple fluents* are fluents which are constants (e.g., a). The set of *fluent terms* is the smallest set satisfying the following conditions: 1 is a fluent term; each fluent is a fluent term; if s and t are fluent terms, then so is $s \circ t$. As the sequences of fluents occurring in a fluent term is not important, we consider $\circ$ to be associative and commutative, and 1 to be a unit element with respect to $\circ$; let $\mathcal{K}_{AC1}$ be the corresponding equational axioms plus the axioms of equality.

There is a one-to-one correspondence between equivalence classes of fluent terms with respect to $\mathcal{K}_{AC1}$ and multisets of fluents as follows: Let t be a fluent term and $\mathcal{M}$ a multiset of fluents in:

$$t^I = \begin{cases} \dot{\emptyset} & \text{if } t = 1, \\ \dot{\{}t\dot{\}} & \text{if } t \text{ is a fluent,} \\ u^I \dot{\cup} v^I & \text{if } t = u \circ v, \end{cases} \quad \text{and} \quad \mathcal{M}^{-I} = \begin{cases} 1 & \text{if } \mathcal{M} \doteq \dot{\emptyset}, \\ s \circ \mathcal{M}'^{-I} & \text{if } \mathcal{M} \doteq \dot{\{}s\dot{\}} \dot{\cup} \mathcal{M}'. \end{cases}$$

3 Conjunctive Planning Problems

Conjunctive planning problems were considered in [8] to relate the fluent calculus to the linear connection method and to linear logic. Here, we consider a slightly simplified version in that we restrict fluents to simple fluents.

A *conjunctive planning problem (CPP)* is a tuple $(\mathcal{I}, \mathcal{G}, \mathcal{A})$, where $\mathcal{I}$ and $\mathcal{G}$ are finite multisets of simple fluents called *initial* and *goal state*, respectively, and $\mathcal{A}$ is a finite set of actions; an *action* is an expression of the form $a : \mathcal{C} \Rightarrow \mathcal{E}$, where a is the *name* of the action and $\mathcal{C}$ and $\mathcal{E}$ are finite multisets of simple fluents called *conditions* and *(immediate) effects*, respectively.

Let $\mathcal{S}$ be a finite multiset of simple fluents; such multisets are called *states* in the sequel. An action $a : \mathcal{C} \Rightarrow \mathcal{E}$ is *applicable* to $\mathcal{S}$ iff $\mathcal{C} \dot{\subseteq} \mathcal{S}$; its *application* leads to the state $(\mathcal{S} \dot{\setminus} \mathcal{C}) \dot{\cup} \mathcal{E}$. A *plan* is a sequence or list of actions; it transforms state $\mathcal{S}$ into $\mathcal{S}'$ if and only if S' is the result of successively applying the actions occurring in the plan to $\mathcal{S}$. A plan is a *solution* to a CPP $(\mathcal{I}, \mathcal{G}, \mathcal{A})$ if and only if it transforms $\mathcal{I}$ into $\mathcal{G}$.

To illustrate CPPs, consider a situation where a man living in an apartment becomes severely ill and calls the ambulance. The ambulance men decide that he needs to undergo treatment in a hospital and carry him on a stretcher to the ambulance. Finally, the ambulance car is driven to the hospital. This problem is considered as a CPP with fluent ill (denoting the ill man), apt (denoting that he is in his apartment), amb (denoting that the patient is in the ambulance car), and hos (denoting that the patient is in the hospital). The action names are c (the ambulance men are carrying the patient to the ambulance car) and d (driving to the hospital). Alltogether we obtain a CPP $(\mathcal{I}, \mathcal{G}, \mathcal{A})$*, where* $\mathcal{I} \doteq \dot{\{}ill, apt\dot{\}}$*,* $\mathcal{G} \doteq \dot{\{}ill, hos\dot{\}}$*, and*

$$\mathcal{A} \doteq \{c : \dot{\{}ill, apt\dot{\}} \Rightarrow \dot{\{}ill, amb\dot{\}},\ d : \dot{\{}ill,\ amb\dot{\}} \Rightarrow \dot{\{}ill, hos\dot{\}}\}.$$

The goal state $\mathcal{G}$ *is reached from the initial state* $\mathcal{I}$ *by applying first c yielding the intermediate state* $\dot{\{}ill, amb\dot{\}}$ *and, thereafter, applying d.*

3.1 Conjunctive Planning Problems in the Fluent Calculus

This subsection is based on [12], where an equational logic programming solution to CPPs was presented. For each action $a : \mathcal{C} \Rightarrow \mathcal{E}$ in a given CPP a fact

$$action(\mathcal{C}^{-I}, a, \mathcal{E}^{-I})$$

is specified; let $\mathcal{K}_A$ be the set of all facts of this form for a given CPP. For the running example we obtain

$$\mathcal{K}_A = \{action(ill \circ apt, c, ill \circ amb), action(ill \circ amb, d, ill \circ hos)\}$$

The applicability of an action is specified by

$$applicable(C \circ Z, A, E \circ Z) \leftarrow action(C, A, E),$$

where Z is a variable which will be used to collect all fluents of a state which are not affected by the application of an action. Causality is specified with the help of a ternary predicate symbol *causes*. Intuitively, $causes(X, P, Y)$ is used to represent that the execution of plan P in state X transforms X into state Y. *causes* is specified inductively on the structure of plans, i.e., lists of actions:

$$causes(X, [\,], X)$$
$$causes(X, [A|P], Y) \leftarrow applicable(X, A, U) \wedge causes(U, P, Y)$$

Let K_C be the set containing the clauses for *applicable* and *causes*. A CPP $(\mathcal{I}, \mathcal{G}, \mathcal{A})$ can now be represented in the fluent calculus by the question of whether

$$\mathcal{K}_A \cup \mathcal{K}_C \cup \mathcal{K}_{AC1} \models (\exists P)\, causes(\mathcal{I}^{-I}, P, \mathcal{G}^{-I}),$$

and SLDE-resolution can be applied to compute an answer substitution for P encoding a solution for the CPP if it is solvable.

Let $\mathcal{FC}_Q$ denote the *fluent calculus representation* of a CPP Q. The following theorem is proven in [8]:

Theorem 1. *Let Q be a CPP. The following statements are equivalent for a plan p:*

1. *p is a solution for Q.*
2. *p is generated by SLDE-resolution for $\mathcal{FC}_Q$.*

3.2 Conjunctive Planning Problems in Petri Nets

Let $Q = (\mathcal{I}, \mathcal{G}, \mathcal{A})$ be a CPP. The Petri net $\mathcal{N}_Q = (\mathcal{P}, \mathcal{T}, \mathcal{F})$ together with the markings $\mathcal{I}$ and $\mathcal{G}$ is the *Petri net representation* of Q, where $\mathcal{P}$ is the set of all simple fluents occurring in Q, $\mathcal{T}$ is the set of all action names occurring in Q, $(p, t) \in_k \mathcal{F}$ if and only if $t : \mathcal{C} \Rightarrow \mathcal{E} \in \mathcal{A}$ such that $p \in_k \mathcal{C}$, and $(t, p) \in_k \mathcal{F}$ if and only if $t : \mathcal{C} \Rightarrow \mathcal{E} \in \mathcal{A}$ such that $p \in_k \mathcal{E}$.

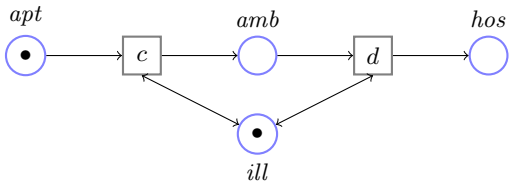

Fig. 1. A Petri net for the ill man problem with initial marking $\dot{\{}apt, ill\dot{\}}$.

One should observe that for each action $a : \mathcal{C} \Rightarrow \mathcal{E}$ in $\mathcal{A}$ we find a transition $a \in \mathcal{T}$ with $\bullet a \doteq \mathcal{C}$ and $a\bullet \doteq \mathcal{E}$. Conversely, whenever a transition t is enabled in $\mathcal{N}_Q$ given a marking $\mathcal{M}$, then there exists an action with name t in $\mathcal{A}$ which is applicable in $\mathcal{M}$.

The question of whether there exists a plan p solving a CPP $(\mathcal{I}, \mathcal{G}, \mathcal{A})$ is the question of whether there exists a firing sequence from $\mathcal{I}$ to $\mathcal{G}$ in $\mathcal{N}_Q$. The Petri net for the running example is depicted in Figure 1. One should observe that $[c, d]$ is a firing sequence leading from the $\dot{\{}apt, ill\dot{\}}$ to $\dot{\{}ill, hos\dot{\}}$.

3.3 Petri Net versus Fluent Calculus Representations

As a first result we extend Theorem 1. Throughout this subsection, let Q be the CPP $(\mathcal{I}, \mathcal{G}, \mathcal{A})$ and $\mathcal{FC}_Q$ and $\mathcal{N}_Q$ be its representations in the fluent calculus and in Petri nets, respectively.

Theorem 2. *The following statements are equivalent for a plan p:*

1. *p is a solution for Q.*
2. *p is generated by SLDE-resolution for $\mathcal{FC}_Q$.*
3. *p is a firing sequence from $\mathcal{I}$ to $\mathcal{G}$ in $\mathcal{N}_Q$.*

Proof. Because of Theorem 1 it suffices to show that the *2.* and *3.* are equivalent. By induction on the number of transitions occurring in p in can be shown that *3.* implies *2.*. The converse can be shown to hold by induction on the number of actions occurring in p.

4 Advanced Planning Problems

In conjunctive planning problems, all conditions of an action are consumed when the action is executed. In this section we extend planning problems to allow preconditions which are only tested when an action is executed but are not consumed and to allow obstacles which prevent an action from being executed even if its conditions are satisfied.

An *advanced conjunctive planning problem (ACPP)* is a tuple $(\mathcal{I}, \mathcal{G}, \mathcal{A})$, where $\mathcal{I}$ and $\mathcal{G}$ are finite multisets of simple fluents called *initial* and *goal state*, respectively, and $\mathcal{A}$ is a finite set of *advanced actions*; an advanced action is an

expression of the form $a : \mathcal{C} \overset{\mathcal{R},\mathcal{O}}{\Longrightarrow} \mathcal{E}$, where a is the *name* of the action, $\mathcal{C}$, $\mathcal{R}$, $\mathcal{O}$ and $\mathcal{E}$ are multisets of simple fluents called *conditions*, *preconditions*, *obstacles*, and *effects*, respectively, and $\mathcal{C} \dot{\cap} \mathcal{E} \doteq \dot{\emptyset}$.

Let $\mathcal{S}$ be a state. An extended action $a : \mathcal{C} \overset{\mathcal{R},\mathcal{O}}{\Longrightarrow} \mathcal{E}$ is *applicable* to $\mathcal{S}$ if and only if $\mathcal{C} \dot{\subseteq} \mathcal{S}$, $\mathcal{R} \dot{\subseteq} \mathcal{S}$, and $\forall e \in_k \mathcal{O}(e \in_j \mathcal{S} \rightarrow j < k)$. Its application yields the state $(\mathcal{S} \setminus \mathcal{C}) \dot{\cup} \mathcal{E}$. If the last condition in the definition of applicability to $\mathcal{S}$ is violated, i.e., if there is an extended action with name a, obstacles $\mathcal{O}$ and $e \in_k \mathcal{O}$, and $e \in_j \mathcal{S}$ such that $j \geq k$, then a is *hindered* in $\mathcal{S}$. *Plans* and *solutions* are defined as before.

To illustrate ACPPs we modify the running example by assuming that the patient was so fat that he did not fit through the appartment door. Hence, the ambulance men cannot carry him to the ambulance car. We introduce an additional fluent fat and obtain the ACPP $(\mathcal{I}, \mathcal{G}, \mathcal{A})$*, where* $\mathcal{I} \doteq \{\!\!\{ill, fat, apt\}\!\!\}$, $\mathcal{G} \doteq \{\!\!\{ill, fat, hos\}\!\!\}$, *and*

$$\mathcal{A} \doteq \{c : \{\!\!\{apt\}\!\!\} \overset{\{\!\!\{ill\}\!\!\},\{\!\!\{fat\}\!\!\}}{\Longrightarrow} \{\!\!\{amb\}\!\!\},\ d : \{\!\!\{amb\}\!\!\} \overset{\{\!\!\{ill\}\!\!\},\dot{\emptyset}}{\Longrightarrow} \{\!\!\{hos\}\!\!\}\}.$$

Obviously, this ACPP cannot be solved as the obstacle fat hinders the application of the action c. By the way, the ill man was later rescued with a help of a mobile cran, which carried him out of his apartment through a window.

4.1 Advanced Planning Problems in Petri Nets

Petri nets were extended by so-called inhibitory arcs, which may by weighted and which increase the modeling power of Petri nets to the level of Turing machines [16,5,4]. We combine inhibitor arcs with so-called test arcs, which were introduced in [2] to allow for places, which may contain real-valued or discrete tokens in order to enable an action, but which are not consumed.

An *advanced Petri net* is a tuple $(\mathcal{P}, \mathcal{T}, \mathcal{F}, \mathcal{H}, \mathcal{L})$, where $(\mathcal{P}, \mathcal{T}, \mathcal{F})$ is a Petri net, $\mathcal{H} \dot{\subseteq} \mathcal{P} \times \mathcal{T}$, and $\mathcal{L} \dot{\subseteq} \mathcal{P} \times \mathcal{T}$; $\mathcal{H}$ and $\mathcal{L}$ are the multiset of *inhibitory* and *test arcs*, respectively. The multiset $\blacktriangledown t$ of *inhibitory places of transition* $t \in \mathcal{T}$ is defined by $p \in_k \blacktriangledown t$ if and only if $p \in \mathcal{P} \wedge (p, t) \in_k \mathcal{H}$. Likewise, the multiset $\blacktriangle t$ of *test places of transition* t is defined as $p \in_k \blacktriangle t$ if and only if $p \in \mathcal{P} \wedge (p, t) \in_k \mathcal{L}$.

Let $\mathcal{N} = (\mathcal{P}, \mathcal{T}, \mathcal{F}, \mathcal{H}, \mathcal{L})$ be an advanced Petri net and $\mathcal{M}$ a marking. $t \in \mathcal{T}$ is *enabled* at $\mathcal{M}$ in $\mathcal{N}$ if and only if $\bullet t \dot{\subseteq} \mathcal{M}$, $\forall p \in \mathcal{P}((p, t) \in_k \mathcal{L} \wedge p \in_j \mathcal{M} \rightarrow k \leq j)$, and $\forall p \in \mathcal{P}((p, t) \in_m \mathcal{H} \wedge p \in_n \mathcal{M} \rightarrow m > n)$. The notions *fire* and *firing sequence* are defined as before.

Let $Q = (\mathcal{I}, \mathcal{G}, \mathcal{A})$ be an ACPP. The Petri net $\mathcal{N}_Q^{\mathcal{A}} = (\mathcal{P}, \mathcal{T}, \mathcal{F}, \mathcal{H}, \mathcal{L})$ together with the markings $\mathcal{I}$ and $\mathcal{G}$ is the *Petri net representation* of Q, where $\mathcal{P}$, $\mathcal{T}$, and $\mathcal{F}$ are defined as in Subsection 3.2, $(p, t) \in_k \mathcal{H}$ if and only if $\exists (t : \mathcal{C} \overset{\mathcal{R},\mathcal{O}}{\Longrightarrow} \mathcal{E}) \in \mathcal{A}$ such that $p \in_k \mathcal{O}$, and $(p, t) \in_k \mathcal{L}$ if and only if $\exists (t : \mathcal{C} \overset{\mathcal{R},\mathcal{O}}{\Longrightarrow} \mathcal{E}) \in \mathcal{A}$ such that $p \in_k \mathcal{R}$. The Petri net for the modified running example is shown in Figure 2.

From this definition we learn that for every action $t : \mathcal{C} \overset{\mathcal{R},\mathcal{O}}{\Longrightarrow} \mathcal{E}$ in $\mathcal{A}$ we find a transition $t \in \mathcal{N}_Q^{\mathcal{A}}$ with $\blacktriangledown t \doteq \mathcal{O}$, $\blacktriangle t \doteq \mathcal{R}$, $\bullet t \doteq \mathcal{C}$, and $t\bullet \doteq \mathcal{E}$. One should observe

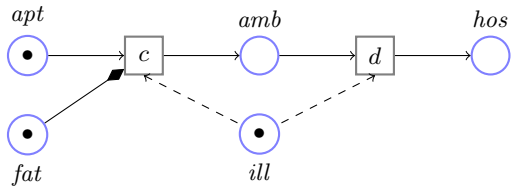

Fig. 2. The advanced Petri net for the modified running example with initial marking. Test arcs are depicted by dashed arrows, inhibitory arcs by arrows with a diamond head.

that the requirements for enabling a transition in $\mathcal{N}_{\mathcal{Q}}^{\mathcal{A}}$ are the requirements for the applicability of an action in $\mathcal{Q}$. Hence, a transition t is enabled at marking $\mathcal{M}$ in $\mathcal{N}_{\mathcal{Q}}^{\mathcal{A}}$ if and only if there exists an action t in $\mathcal{Q}$ with t being applicable in $\mathcal{M}$.

4.2 Advanced Planning Problems in the Fluent Calculus

To maintain the additional features of ACPPs several new facts and rules are added to the fluent calculus representation introduced in Subsection 3.1. For each advanced action $a : \mathcal{C} \stackrel{\mathcal{R},\mathcal{O}}{\Longrightarrow} \mathcal{E}$ and each obstacle $o \in_k \mathcal{O}$ the fact

$$inhib(\overbrace{o \circ \ldots \circ o}^{k \text{ times}}, a)$$

is added to $\mathcal{K}_A$. In addition, for each advanced action $a : \mathcal{C} \stackrel{\mathcal{R},\mathcal{O}}{\Longrightarrow} \mathcal{E}$ the fact

$$precon(\mathcal{R}^{-I}, A)$$

is added to $\mathcal{K}_A$; it is used in the (modified) definition of *applicable* to test whether all preconditions are met. For our modified running example we obtain

$$\begin{aligned}\mathcal{K}_A = \{\ & action(apt, c, amb),\ action(amb, d, hos),\\ & inhib(fat, c),\ precon(ill, c),\ precon(ill, d)\ \}.\end{aligned}$$

The rule

$$hinder(X \circ Z, A) \leftarrow inhib(X, A)$$

is added to $\mathcal{K}_C$ to prohibit the application of action A whenever sufficiently many obstacles X are present in a state $X \circ Z$. The definition of *applicable* in $\mathcal{K}_C$ is modified to

$$\begin{aligned}applicable(C \circ Z, A, E \circ Z) \leftarrow\ & action(C, A, E) \wedge\\ & precon(R, A) \wedge R \circ Y \approx C \circ Z \wedge\\ & \neg hinder(C \circ Z, A)\end{aligned}$$

where the subgoal $R \circ Y \approx C \circ Z$ is used to check whether the preconditions R of action A are satisfied in state $C \circ Z$. As the equality predicate $\approx$ is now used explicitly as a subgoal, the axiom of reflexivity

$$X \approx X$$

must be added to $\mathcal{K}_C$, effectively forcing the AC1-unification of the left-hand and the right-hand side of the subgoal $R \circ Y \approx C \circ Z$.

Let $\mathcal{K}_A^{\mathcal{A}}$ and $\mathcal{K}_C^{\mathcal{A}}$ be the modified sets of clauses for a given ACPP $(\mathcal{I}, \mathcal{G}, \mathcal{A})$. The ACPP can now be presented in the fluent calculus by the question of whether

$$\mathcal{K}_A^{\mathcal{A}} \cup \mathcal{K}_C^{\mathcal{A}} \cup \mathcal{K}_{AC1} \models (\exists P)\ \mathit{causes}(\mathcal{I}^{-I}, P, \mathcal{G}^{-I}),$$

and SLDENF-resolution can be applied to compute an answer substitution for P, if existing [13]. SLDENF-resolution is sound [17], but it is only shown to be complete if the completion of $\mathcal{K}_A^{\mathcal{A}} \cup \mathcal{K}_C^{\mathcal{A}} \cup \mathcal{K}_{AC1}$ is satisfiable and SLDENF-derivations neither flounder nor are infinite [13].

Lemma 3. *The completion of $\mathcal{K}_A^{\mathcal{A}} \cup \mathcal{K}_C^{\mathcal{A}} \cup \mathcal{K}_{AC1}$ is satisfiable.*

Proof. By construction of a model for the completion $\mathcal{K}_A^{\mathcal{A}} \cup \mathcal{K}_C^{\mathcal{A}} \cup \mathcal{K}_{AC1}$.

Regarding the question of whether SLDENF-derivations flounder or are infinite we observe that the definition of *causes* is recursive in the second argument, which is a list. If the length of this list is known in advance and the first argument of *causes* is a ground fluent term (which holds by the definition of a planning problem), then SLDENF-derivations neither flounder nor are infinite.

Proposition 4. *Let s be a ground fluent term and a the name of an action. Then, each SLDENF-derivation of $\leftarrow \mathit{hinder}(s, a)$ is finite.*

Proof. Follows immediately from the definition of *hinder* and *inhib*.

Proposition 5. *No SLDENF-derivation of $\leftarrow \mathit{causes}(\mathcal{I}^{-I}, [A_1, \ldots, A_n], \mathcal{G}^{-I})$ flounders or is infinite.*

Proof. By induction on n.

Based on this result we must refine the fluent calculus representation of an ACPP to the question of whether

$$\mathcal{K}_A^{\mathcal{A}} \cup \mathcal{K}_C^{\mathcal{A}} \cup \mathcal{K}_{AC1} \models (\exists A_1, \ldots, A_n)\ \mathit{causes}(\mathcal{I}^{-I}, [A_1, \ldots, A_n], \mathcal{G}^{-I}).$$

and iteratively increase n in the search for a solution of the planning problem.

Finally, we show that *hinder* prevents actions from being applicable:

Proposition 6. *There are enough obstacles in a state $\mathcal{S}$ to hinder an advanced action a if and only if there is an SLDENF-resolution proof of $\leftarrow \mathit{hinder}(\mathcal{S}^{-I}, a)$.*

Proof. Follows from the definitions of *hinder* and *inhib*.

4.3 Petri Nets versus Fluent Calculus Representations

Throughout this subsection, let Q be the ACPP $(\mathcal{I}, \mathcal{G}, \mathcal{A})$, $\mathcal{FC}_Q^{\mathcal{A}}$ and $\mathcal{N}_Q^{\mathcal{A}}$ be its representations in the advanced fluent calculus and the advanced Petri nets, respectively, and p be the plan $[a_1, \ldots, a_n]$.

Theorem 7. *The following statements are equivalent for a plan p:*

1. *p is a solution for Q.*
2. *p is generated by SLDENF-resolution for $\mathcal{FC}_Q^{\mathcal{A}}$.*
3. *p is a firing sequence from $\mathcal{I}$ to $\mathcal{G}$ in $\mathcal{N}_Q^{\mathcal{A}}$.*

Proof. The theorem is obtained if we can prove that *1.* implies *2.*, *2.* implies *3.*, and *3.* implies *1.* These implications can be shown by inductions on the length of the plan p, on the length of the SLDENF-resolution refutation, and on the length of the firing sequence, respectively.

5 Discussion

In this paper we have shown that there is a close correspondence between Petri nets and the fluent calculus for conjunctive planning problems. This correspondence is preserved if we extended Petri nets and the fluent calculus by test and inhibitory arcs. The correspondence can be even further extended to planning problems with fluents containing real values as investigated in [15]. In [2], it was shown that Petri nets can be combined with Bayes nets via real-valued fluents, and, hence, it should now be possible to combine the fluent calculus and Bayes nets. However, this needs to be rigourously investigated in the near future.

Whereas in this paper we were computing answer substitutions by SLDE- and SLDENF-resolution in the fluent calculus, we also like to invesigate the corresponding fixpoint characterization of the fluent calculus. This is the obvious next step in order to combine the human reasoning approach mentioned in the introduction with reasoning about actions and causality in the fluent calculus. Finally, the ultimate goal is a connectionist realization of the combined approach within the core method [11,1].

Acknowledgements We would like to thank Bertram Fronhöfer and Christoph Wernhard for many fruitful discussions and the anonymous referees for there comments.

References

1. S. Bader and S. Hölldobler. The core method: Connectionist model generation. In S. Kollias et. al., editor, *Proceedings of the 16th International Conference on Artificial Neural Networks (ICANN)*, volume 4132 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2006.
2. L. Barrett. *An Architecture for Structured, Concurrent, Real-time Action.* PhD thesis, Computer Science Division, University of California at Berkeley, 2010.

3. R. M. J. Byrne. Suppressing valid inferences with conditionals. *Cognition*, 31:61–83, 1989.
4. R. David and H. Alla. On hybrid Petri nets. *Discrete Event Dynamic Systems*, 11(1-2):9–40, 2001.
5. J. Desel and W. Reisig. *Lectures on Petri Nets I: Basic Models*, volume 1491 of *LNCS*, chapter Place/Transition Petri Nets, pages 122–173. Springer, 1998.
6. E.-A. Dietz, S. Hölldobler, and M. Ragni. A computational logic approach to the suppression task. In N. Miyake, D. Peebles, and R. P. Cooper, editors, *Proceedings of the 34th Annual Conference of the Cognitive Science Society*, pages 1500–1505. Cognitive Science Society, 2012.
7. E.-A. Dietz, S. Hölldobler, and M. Ragni. A computational logic approach to the abstract and the social case of the selection task. In *Proceedings Eleventh International Symposium on Logical Formalizations of Commonsense Reasoning*, 2013.
8. G. Große, S. Hölldobler, and J. Schneeberger. Linear deductive planning. *Journal of Logic and Computation*, 6(2):233–262, 1996.
9. S. Hölldobler. *Foundations of Equational Logic Programming*, volume 353 of *Lecture Notes in Artificial Intelligence*. Springer, Berlin, 1989.
10. S. Hölldobler and C. D. P. Kencana Ramli. Logic programs under three-valued Łukasiewicz's semantics. In P. M. Hill and D. S. Warren, editors, *Logic Programming*, volume 5649 of *Lecture Notes in Computer Science*, pages 464–478. Springer Berlin Heidelberg, 2009.
11. S. Hölldobler and C. D. P. Kencana Ramli. Logics and networks for human reasoning. In C. Alippi, Marios M. Polycarpou, Christos G. Panayiotou, and Georgios Ellinasetal, editors, *Artificial Neural Networks – ICANN*, volume 5769 of *Lecture Notes in Computer Science*, pages 85–94. Springer Berlin Heidelberg, 2009.
12. S. Hölldobler and J. Schneeberger. A new deductive approach to planning. *New Generation Computing*, 8:225–244, 1990.
13. S. Hölldobler and M. Thielscher. Computing change and specificity with equational logic programs. *Annals of Mathematics and Artificial Intelligence*, 14:99–133, 1995.
14. P.N. Johnson-Laird. *Mental Models: Towards a Cognitive Science of Language, Inference, and Consciousness*. Cambridge University Press, Cambridge, 1983.
15. F. Jovan. Planning problems in Petri nets and fluent calculus. Master's thesis, TU Dresden, Faculty of Computer Science, 2014.
16. T. Murata. Petri nets: Properties, analysis and applications. In *Proceedings of the IEEE*, volume 77, pages 541–580, 1989.
17. J. C. Shepherdson. SLDNF–resolution with equality. *Journal of Automated Reasoning*, 8:297–306, 1992.
18. K. Stenning and M. van Lambalgen. *Human Reasoning and Cognitive Science*. MIT Press, 2008.
19. M. Thielscher. Introduction to the fluent calculus. *Electronic Transactions on Artificial Intelligence*, 2:179–192, 1998.