SCOPE: A Situation Calculus Ontology of Petri Nets

Xing TAN [1]
Semantic Technologies Laboratory, Department of Mechanical and Industrial Engineering, University of Toronto

Abstract. By axiomatizing the semantics of Petri nets as a first-order (mostly) formal ontology called SCOPE, we propose in this paper a framework for the analysis of the structural and dynamical properties of Petri nets. More precisely, SCOPE is built as a Basic Action Theory in Reiter's version of situation calculus. In addition, we show the satisfiability of SCOPE, by virtue of the Relative Satisfiability Theorem. Fundamental structural and dynamical properties of Petri nets described in SCOPE are also presented, with two example uses of SCOPE given.

Keywords. Petri Nets, Situation Calculus, Basic Action Theory, Ontology, Model Characterization

Introduction

Petri nets as a powerful modeling tool have long been used to describe complex dynamical systems that involve concurrency, nondeterminism, asynchronism, and intense interactions between events and agents in these systems. The simple formalism that Petri nets offer has enabled unambiguous graphical representations of Petri nets and facilitated studies on the properties of Petri nets with mathematical rigor.

One recent application of Petri nets is within the context of the semantic Web, where, in contrast to the current Web, the semantics of the content and capability are well-defined such that they are machines-interpretable, enabling automation of a variety of different operations currently performed by human-beings in essence. In practice, Petri nets have shown to be quite handy in providing the operational semantics of certain class of Web services, i.e., Web-accessible service programs or devices. For example, in [4], in order to automatically describe, compose, simulate, and verify the Web service compositions, a Petri net formalism is employed to describe the operational semantics of DARPA Agent Markup Language-Services (DAML-S), a logic language describing Web services, after the descriptive semantics of DAML-S is first specified by situation calculus. Our thesis is that, Petri nets should have much broader applicability for the semantic Web. The rationale for this claim is that the logic action language situation calculus, particularly its variant as comprehensively studied by Reiter and et al. ([9], [10]), provides a simple but mathematically rigorous paradigm for axiomatization of dynamical systems, making it presumably an easy task to develop a situation-calculus-

[1] E-mail: xtan@mie.utoronto.ca.

based Petri nets action theory. As such, we propose in this paper SCOPE, a situation-calculus-based ontology of Petri nets.

SCOPE is introduced in Section 3 of this current paper. It adopts the simplicity of original Petri nets: it contains only one action type, namely, firing of transitions; whereas only one function value, namely the number of tokens at places, is subject to the change upon actions. We argue in this section the explicit applicability of certain assumption (the *causal completeness assumption*, to be precise) for the development of SCOPE. By proving that SCOPE satisfies both the unique formula requirement and the function consistency property, we show in addition that SCOPE is a Basic Action Theory, which qualifies the applicability of the Relative Satisfiability Theorem on SCOPE in Section 4.

In Section 4, we characterize a class of combinatorial structures and prove both the satisfiability theorem and the axiomatizability theorem for the class; that is, respectively, every structure in the class is a model for some subset of SCOPE axioms (including unique name axioms for actions and all axioms that are uniform in the initial state) and every model for the subset is isomorphic to some structure in the class. By applying the Relative Satisfiability Theorem, we also have that such a model can be extended to a model for all SCOPE axioms. In other words, SCOPE is satisfiable.

Other sections of this paper are as follows. Section 1 and Section 2 provide brief introductions to Petri nets and Reiter's situation calculus, respectively. Section 5 further provides a set of axiomatization to the structure constraints and mathematical properties of Petri nets and includes two examples of SCOPE use cases. Section 6 summarizes the paper.

1. Petri Nets

Petri nets and their markings are defined in Section 1.1, whereas several dynamical properties of Petri nets and structurally restricted subclasses of Petri nets are introduced in Section 1.2.

1.1. Basic Definitions

Definition 1 *A Petri net (PN) is a pair (N, M_0) where N is a triple (P, T, F) such that*

- *P is a finite set of node elements called places;*
- *T is a finite set of node elements called transitions;*
- *and $F \subseteq (P \times T) \cup (T \times P)$ consists of ordered pairs;*

and M_0 is the initial marking, a mapping in the form $M\colon P \rightarrow \mathcal{N}$, indicating the initial assignment of a non-negative integer k to each place p in P. (In this case, we say that the place p is marked with k tokens.)

In general, any marking M for N in PN is defined as a vector $(M(p_1), \ldots, M(p_m))$ where $p_1, \ldots, p_m$ is an enumeration of P and $M(p_i)$ tokens are assigned to node p_i, for all i such that $1 \leq i \leq m$.

The elements in $P \cup T$ are generically called nodes of PN. Given a node $u \in PN$, the set ${}^\bullet u = \{v | (v, u) \in F\}$ is the pre-set of u, where each v is the input of u, and the set $u^\bullet = \{v | (u, v) \in F\}$ is the post-set of u, where each v is the output of u. At any marking M, a place p is marked if $M(p) > 0$. A transition t is enabled in M if every place in ${}^\bullet t$ is marked in M. Enabled transition in M can occur (fire), leading to the successor marking M'.

Definition 2 *A marking transition from M to M' due to the firing of t (written as $M \overset{t}{\Longrightarrow} M'$) is defined by*

$$M'(p) = \begin{cases} M(p) - 1 & \textit{if } p \in {}^\bullet t \textit{ and } p \notin t^\bullet \\ M(p) + 1 & \textit{else if } p \notin {}^\bullet t \textit{ and } p \in t^\bullet \\ M(p) & \textit{otherwise} \end{cases}$$

for every place p.

Graphically, a Petri Net PN can be represented by a bipartite graph, where each place is represented by a circle, each transition is represented by a rectangle, the flow relation of the Petri net F is represented by arcs from places to transitions or from transitions to places, and k black dots will be placed into the rectangle for place p if it is marked with k tokens at M. Figure 1 (A) is a PN example where transitions T1 and T2 can be interpreted as *switch on* and *switch off*, respectively.

1.2. Petri Nets Properties

The concept of reachable marking is essential to the description and analysis of all properties (such as reachability, k-boundedness, and reversibility, as defined below) of Petri nets.

Definition 3 (reachable marking) *The transition (firing) sequence $\sigma = t_1, ..., t_m$ is a sequence of transition nodes in T; a σ is an occurrence sequence of a given Petri net $PN = (N, M_0)$ if $M_0 \overset{t_1}{\Longrightarrow} M_1$, ..., $M_{m-1} \overset{t_m}{\Longrightarrow} M_m$ (written as $M \overset{\sigma}{\Longrightarrow} M_m$). Also, we write $M \overset{*}{\Longrightarrow} M'$ and call M' is reachable from M if there exists a firing sequence σ such that $M \overset{\sigma}{\Longrightarrow} M'$. The set of all markings reachable from M is denoted by $R(N, M)$.*

Definition 4 (Reachability) *The reachability problem for Petri nets is the problem of finding, given a Petri net (N, M_0) and a marking M in it, if $M \in R(N, M_0)$.*

Definition 5 (k-boundedness) *Given a Petri net (N, M_0), the k-boundedness problem involves checking whether the number of tokens in each place does not exceed the integer k for any marking reachable from M_0. A 1-bounded Petri net is also said to be "1-safe".*

Definition 6 (Liveness) *A Petri net (N, M_0) is live if, for any marking $M \in R(N, M_0)$, there exists another marking M', such that $M \overset{*}{\Longrightarrow} M'$, i.e., M' is reachable from M.*

Definition 7 (Reversibility/Cycleness) *A Petri net (N, M_0) is reversible if M_0 is reachable from each $M \in R(N, M_0)$.*

Definition 8 (Coverability) *A Petri net (N, M_0) is coverable if there exists a marking $M' \in R(N, M_0)$ such that for any $p \in P$, $M'(p) \geq M_0(p)$.*

Definition 9 (Persistency) *A Petri net (N, M_0) is persistent if, for any $t_1 \in T$ and $t_2 \in T$, which are both enabled at any $M \in R(N, M_0)$, t_2 should still be enabled in M' where $M \overset{t_1}{\Longrightarrow} M'$.*

Given that Petri nets as a modeling language has rich expressive power, evaluating properties of a Petri net is often computationally expensive, if possible. However, the difficulty is largely reduced if we are dealing with certain subclasses of Petri Nets where their graphical structures are highly constrained. The properties of the subclasses of Petri nets defined as follows have been thoroughly studied.

Definition 10 *A Petri net $PN = (N, M_0)$, where N is (P, T, F), is an S-system if $|^{\bullet}t| = |t^{\bullet}| = 1$ for every $t \in T$.*

Definition 11 *A Petri net $PN = (N, M_0)$, where N is (P, T, F), is a T-system if $|^{\bullet}p| = |p^{\bullet}| = 1$ for every $p \in P$.*

Definition 12 *A Petri net $PN = (N, M_0)$, where N is (P, T, F), is a conflict-free system if $p^{\bullet} \subseteq {}^{\bullet}p$ for every place p with more than one output transition.*

Definition 13 *A Petri net $PN = (N, M_0)$, where N is (P, T, F), is a free-choice system if $(p, t) \in F$ implies ${}^{\bullet}t \times p^{\bullet} \subseteq F$ for every place p and every transition t.*

Later we will show, in Section 5.1, that all of these above properties or subclasses can be represented easily in SCOPE.

2. Reiter's Situation Calculus

2.1. Brief Introduction

The situation calculus is a logical language for representing changes upon actions in a dynamical domain; it is first proposed by McCarthy and Hayes in 1969 [6]. The language $\mathcal{L}$ of situation calculus as stated by [10] is a second-order many-sorted language with equality.

Three disjoint sorts: *action*, *situation*, *object* (for everything else in the specified domain) are included in the language $\mathcal{L}$. For example, *rain* denotes the act of raining, and $putdown(x, y)$ denotes the act of object x putdown y on the ground. A *situation* characterizes a sequence of actions in the domain. The constant situation S_0 is to denote the empty sequence of actions, whereas the function symbol *do* is introduced to construct the term $do(a, s)$, denoting the successor situation after performing action a (such as, in a weather simulation scenario, *rain*) in situation s. The situation term $do(sunshine, do(rain, s))$ denotes the situation resulting from first $rain$ and then $sunshine$, which distinguishes itself from the other situation term $do(rain, do(sunshine, s))$.

The binary predicate $\sqsubset$ specifies the order between situations. For example, $s \sqsubset s'$ stands for that the situation s' can be reached by performing one or several actions from s. $s \sqsubseteq s'$ is an abbreviation of $s \sqsubset s' \vee s = s'$. In addition, a predicate $Poss(a, s)$ is applied to specify the legality of performing action a in situation s, For example, $Poss(rain, s) \supset heavyCloudy$ says that it is possible to rain only if the sky is with heavy cloud.

In a particular domain, the language might contain situation independent relations, like $matchLocation(Toronto)$, and situation independent functions, like $size(Plot2)$.

However, in many of the more interesting cases, the values of relations and functions change between situations; accordingly, a relational fluent, or a functional fluent, in $\mathcal{L}$ is defined as a predicate, or a function, respectively, whose last argument is always a situation (e.g., $captain(John, do(catchFever, S_0))$ is a relational fluent, whereas $weight(John, do(recover, s))$ is a functional fluent).

A particular class of situation calculus theories called the Basic Action Theories is specified in [10] (also briefly presented in the subsequent section), where nice computational properties holds for operating regression on sentences satisfying certain conditions. For situation calculus at an introductory level, readers are referred to [2] and [1], where each includes a chapter of situation calculus. In-depth discussion is included in [10].

2.2. Basic Action Theories

The foundational axioms captures the concept of situation tree in a domain, whereas the Basic Action Theories are special situation calculus theories where nice theoretical and computational properties, such as the Relative Satisfiability Theorem, hold. (see [10] for more details.)

Definition 14 *The set $\mathcal{D}_f$ consists of four foundational axioms as follows*

$$do(a_1, s_1) = do(a_2, s_2) \supset a_1 = a_2 \wedge s_1 = s_2 \tag{1}$$

$$(\forall P).P(S_0) \wedge (\forall a, s)\big(P(s) \supset P(do(a, s))\big) \supset (\forall s)P(s) \tag{2}$$

$$\neg s \sqsubset S_0 \tag{3}$$

$$s \sqsubset do(a, s') \equiv s \sqsubseteq s' \tag{4}$$

Definition 15 *A basic action theory $\mathcal{D}$ is a collection of several sets of axioms as follows:*

$$\mathcal{D} = \mathcal{D}_f \cup \mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0}$$

where

- *$\mathcal{D}_f$ is the set of foundational axioms;*
- *$\mathcal{D}_{ss}$ is the set of successor state axioms for functional and relational fluents and*
  - ∗ *each relational fluent axiom is of the form*

    $$F(x_1, \ldots, x_n, do(a, s)) \equiv \Phi_F(x_1, \ldots, x_n, a, s)$$

    where $\Phi_F(x_1, \ldots, x_n, a, s)$ is uniform in s;
  - ∗ *each functional fluent axiom is of the form*

    $$f(x_1, \ldots, x_n, do(a, s)) = y \equiv \Phi_f(x_1, \ldots, x_n, y, a, s)$$

    where $\Phi_f(x_1, \ldots, x_n, y, a, s)$ is uniform in s;
  - ∗ *each functional fluent satisfies the consistency property (defined below);*
- *$\mathcal{D}_{ap}$ is the set of precondition axioms of the form*

  $$Poss(A(x_1, \ldots, x_n), s) \equiv \Pi_A(x_1, \ldots, x_n, s),$$

  where A is an n-ary action function symbol and $\Pi_A(x_1, \ldots, x_n, s)$ is uniform in s;

- $\mathcal{D}_{una}$ *is the action unique name axioms;*
- $\mathcal{D}_{S_0}$ *is a set of first-order sentences that are uniform in* S_0.

Definition 16 *A functional fluent f satisfies the consistency property if*
$\mathcal{D}_{una} \cup \mathcal{D}_{S_0} \models$
$(\forall \vec{x}).(\exists y)\phi_f(\vec{x}, y, a, s) \wedge \big((\forall y, y').\phi_f(\vec{x}, y, a, s) \wedge \phi_f(\vec{x}, y', a, s) \supset y = y'\big)$
and $f(\vec{x}, do(a, s)) = y \equiv \phi_f(\vec{x}, y, a, s)$ is the corresponding successor state axioms for f in $\mathcal{D}_{ss}$ (Definition 4.4.5 in [10]) .

The concept of the uniform formulas is formally defined in Definition 4.4.1 of [10]. Intuitively, the uniformity requirement for the precondition axioms and successor state axioms assure that the qualification of an action and the properties in the successor situation, respectively, are determined entirely by the current situation s.

One important theoretical result with respect to the Basic Action Theories is stated as follows. In Section 4, we will show its technical importance towards proving the satisfiability of SCOPE.

Theorem 1 (Relative Satisfiability) *A basic action theory $\mathcal{D}$ is satisfiable iff $\mathcal{D}_{una} \cup \mathcal{D}_{S_0}$ is (Theorem 4.4.7 in [10], and see [9] for the proof).*

3. SCOPE

SCOPE is formally defined in Sectin 3.1. In Section 3.2, we show that indeed SCOPE constructs a Basic Action Theory.

3.1. The Ontology

SCOPE is defined as a logical theory $\mathcal{S}$, which consists of several sets of axioms as follows:

$$\mathcal{S} = \mathcal{D}_f \cup \mathcal{S}_{ap} \cup \mathcal{S}_{ss} \cup \mathcal{S}_{una} \cup \mathcal{S}_{S_0}$$

where

- $\mathcal{D}_f$ is the foundational axioms;
- $\mathcal{S}_{ap}$ (Action Precondition Axioms)

  $$Poss(fire(t), s) \equiv pre(p, t) \supset NumTkns(p, s) \geq 1$$

- $\mathcal{S}_{ss}$ (Successor State Axioms)
  $NumTkns(p, do(a, s)) = n \equiv$
  $\gamma_f(p, t, n, a, s) \vee (NumTkns(p, s) = n \wedge \neg(\exists n')\, \gamma_f(p, t, n', a, s))$, where [2]
  $\gamma_f(p, t, n, a, s) \overset{def}{=} \gamma_{f_a}(p, t, n, a, s) \vee \gamma_{f_b}(p, t, n, a, s)$,
  $\gamma_{f_a}(p, t, n, a, s) \overset{def}{=} ((\exists t).pre(t, p) \wedge \neg post(t, p) \wedge n = NumTkns(p, s) + 1 \wedge a = fire(t))$,
  $\gamma_{f_b}(p, t, n, a, s) \overset{def}{=} ((\exists t).pre(p, t) \wedge \neg post(p, t) \wedge n = NumTkns(p, s) - 1 \wedge a = fire(t))$.

[2] $\gamma_f(p, t, n, a, s)$, $\gamma_{f_a}(p, t, n, a, s)$, and $\gamma_{f_b}(p, t, n, a, s)$ are simply abbreviations.

- $\mathcal{S}_{una}$ (Unique Name Axioms) [3]

$$fire(t_1) = fire(t_2) \supset (t_1 = t_2)$$

- $\mathcal{S}_{S_0}$

  $*$ $pre(p,t) \equiv post(t,p)$, which means that the place p is the input of the transition t if and only if t is the output of p;
  $*$ for each arc from transition t to place p, define $pre(p,t)$; similarly, define $pre(t,p)$;
  $*$ for each place p with initial marking k, define $NumTkns(k, S_0) = k$;

While modeling dynamical systems usually is a complicated task involving identifying complex networks of causal influences, axiomatizing Petri nets is performed directly on the abstract mathematical models and thus is free of this particular type of difficulty. In particular, we now address the relationship between the set of effect axioms $\mathcal{S}_{ef}$ for SCOPE and $\mathcal{S}_{ss}$ in SCOPE with respect to the Causal Completeness Assumption. In short, $\mathcal{S}_{ef}$ is not part of $\mathcal{S}$ but is entailed by $\mathcal{S}$.

Definition 17 *The set of effect axioms $\mathcal{S}_{ef}$ in SCOPE are defined as follows*

$$pre(t,p) \wedge \neg post(t,p) \supset NumTkns(p, do(fire(t), s)) = NumTkns(p, s) + 1,$$

$$pre(p,t) \wedge \neg post(p,t) \supset NumTkns(p, do(fire(t), s)) = NumTkns(p, s) - 1,$$

which is logically equivalent to

$$\gamma_f(p,t,n,a,s) \supset NumTkns(p,t,do(a,s)) = n. \tag{5}$$

Given the clear definition on the transition rules in Petri nets, we know that we can make a causal completeness assumption (claim) from Equation 5; that is, it includes all the conditions under which the only action in the domain *fire* can cause the fluent *NumTkns* to have a value *n* in the successor situation. The Explanation Closure Axiom corresponding to this claim is

$$NumTkns(p,t,do(a,s)) \neq NumTkns(p,t,s) \supset (\exists t)\gamma_f(p,t,n,a,s) \tag{6}$$

Theorem 2 *Equation 5 and Equation 6 together, are logically equivalent to the only successor state axiom in SCOPE*

$$NumTkns(p, do(a,s)) = n \equiv$$

$$\gamma_f(p,t,n,a,s) \vee (NumTkns(p,s) = n \wedge \neg(\exists n')\, \gamma_f(p,t,n',a,s))$$

Proof. see [10] for a proof on general cases. □

It is worth noting that the consistency of $\mathcal{S}_{ef}$ for SCOPE relies on the holding of the sentence

$$\neg(\exists p,t,n,n',a,s).\gamma_f(p,t,n,a,s) \wedge \gamma_f(p,t,n',a,s) \wedge n \neq n' \tag{7}$$

[3]In general, for two distinct action a_1 and a_2, it is required in $\mathcal{S}_{una}$ that $a_1(\vec{x}) \neq a_2(\vec{y})$, but here they are not included as in SCOPE we have only one action "fire".

which in SCOPE states that the action *fire* cannot assign two different values to the fluent *NumTkns* to the successor situation whereas the sentence is simply entailed by applying $\mathcal{S}_{una}$.

3.2. A Basic Action Theory

Theorem 3 *SCOPE constructs a Basic Action Theory.*

Proof. It can be easily verified that the uniform requirement on the current situation s is satisfied in the precondition axioms, the successor state axioms, and the initial condition axioms of SCOPE.

The ontology contains one successor state axiom, for the only functional fluent $NumTkns$. Now we show the function consistency property as defined in Section 2 is satisfied with respect to the $NumTkns$.

The only successor state axiom in $\mathcal{S}_{ss}$ can be written in the form

$$NumTkns(p, do(a, s)) = n \equiv \phi_f(p, t, y, a, s)$$

where

$$\phi_f(p, t, y, a, s) \overset{def}{=} \gamma_f(p, t, n, a, s) \vee (NumTkns(p, s) = n \wedge \neg(\exists n')\, \gamma_f(p, t, n', a, s)).$$

Similar to the proof for Equation 7, we have

$$\mathcal{D}_{una} \models \ (\forall p, t, n, n', a, s).\ \gamma_f(p, t, n, a, s) \wedge \gamma_f(p, t, n', a, s) \supset n = n',$$

and from the definition of ϕ_f, we have

$$\mathcal{D}_{una} \models \ (\forall p, t, n, n', a, s).\ \phi_f(p, t, n, a, s) \wedge \phi_f(p, t, n', a, s) \supset n = n'.$$

Based on the definition of $\mathcal{S}_{S_0}$, we have

$$\mathcal{S}_{S_0} \models (\forall p, t)(\exists n)\phi_f(p, t, n, a, S_0)$$

Thus,

$$(\forall p, t, n, n', a, s)\ \phi_f(p, t, n, a, s) \wedge \phi_f(p, t, n', a, s) \supset n = n' \cup$$

$$(\forall p, t)(\exists n)\phi_f(p, t, n, a, s);$$

that is, informally, for every place p, there exists a unique value of token numbers n at initial situation S_0, and this uniqueness is maintained at every successor situations.

Hence, $\mathcal{D}_{una} \cup \mathcal{D}_{S_0} \models$
$(\forall p, t).(\exists n)\phi_f(p, t, n, a, s) \wedge (\forall n, n', a, s)\big(\phi_f(p, t, n, a, s) \wedge \phi_f(p, t, n', a, s) \supset n = n'\big)$. $\square$

4. Satisfiability of SCOPE

The bipartite incidence structure defined as follows is the building block for specifying the models of SCOPE.

Definition 18 *A bipartite incidence structure is a tuple* $\mathbf{B} = (\Omega_1, \Omega_2, \boldsymbol{in})$*, where* Ω_1, Ω_2 *are disjoint sets such that* $\boldsymbol{in} \subseteq (\Omega_1 \times \Omega_2)$*. Two elements that are related by **in** are called incident.*

Definition 19 *A Petri net structure* $\mathbf{P}^{ini}$ *is defined to include a 7-tuple in the form* $\langle P, T, F, \mathcal{N}, \boldsymbol{PtoT}, \boldsymbol{TtoP}, \boldsymbol{NumOfTkns}\rangle$ *where*

- P*,* T*,* F*,* $\mathcal{N}$ *are disjoint sets, presumably standing for places, transitions, fire actions, and natural numbers, respectively;*
- $\langle P, T, \boldsymbol{PtoT}\rangle$*,* $\langle T, P, \boldsymbol{TtoP}\rangle$*, and* $\langle P, \mathcal{N}, \boldsymbol{NumOfTkns}\rangle$ *are bipartite incidence structures;*

In addition, $\mathbf{P}^{ini}$ *contains,*

- *the unary function **fire*** $: \boldsymbol{T} \rightarrow \boldsymbol{F}$ *is defined such that, for any* $\langle t_1, f_1\rangle$ *and* $\langle t_2, f_2\rangle$ *in **fire**,* $f_1 = f_2$ *only if* $t_1 = t_2$*;*
- *the relation set **pre*** $: (\boldsymbol{P} \cup \boldsymbol{T}) \times (\boldsymbol{P} \cup \boldsymbol{T})$ *is defined such that, for any two nodes* $n_1, n_2 \in (\boldsymbol{P} \cup \boldsymbol{T})$*,* $\langle n_1, n_2\rangle \in \boldsymbol{pre}$ *iff* $\langle n_1, n_2\rangle \in \boldsymbol{PtoT}$ *or* $\langle n_1, n_2\rangle \in \boldsymbol{TtoP}$*;*
- *the relation set **post*** $: (\boldsymbol{P} \cup \boldsymbol{T}) \times (\boldsymbol{P} \cup \boldsymbol{T})$ *is defined such that* $\langle n_1, n_2\rangle \in \boldsymbol{post}$ *iff* $\langle n_2, n_1\rangle \in \boldsymbol{pre}$*.*

Theorem 4 *Any structure in* $\mathbf{P}^{ini}$ *is a model of* $\mathcal{S}_{una\_S_0}$*.*

Proof. (Sketch.)
Suppose we have a $\mathbf{P} \in \mathbf{P}^{ini}$ ($\mathbf{P}^{ini}$ is obviously non-empty), it is easy to verify that each axiom in $\mathcal{S}_{una\_S_0}$ is satisfied by $\mathbf{P}$, that is:
$\mathbf{P} \models fire(t_1) = fire(t_2) \supset (t_1 = t_2)$ (by definition of **fire**)
$\mathbf{P} \models pre(p, t) \equiv post(t, p)$ (by definition of **pre** and **post**)
$\mathbf{P} \models pre(n_1, n_2)$ for each $pre(n_1, n_2) \in S_{S_0}$ (by **pre**),
$\mathbf{P} \models NumTkns(k, S_0) = k$ where $NumTkns(k, S_0) = k \in S_{S_0}$ (by definition of $\langle P, \mathcal{N}, \mathbf{NumOfTkns}\rangle$). □

Theorem 5 *Any model* $\mathcal{M}$ *of* $\mathcal{S}_{una\_S_0}$ *is isomorphic to a structure in* $\mathbf{P}^{ini}$*.*

Proof. (Sketch.) Let $\mathcal{M}$ be a many typed model of $\mathcal{S}_{una\_S_0}$, it is required that
$\mathcal{M} \models fire(t_1) = fire(t_2) \supset (t_1 = t_2)$
$\mathcal{M} \models pre(p, t) \equiv post(t, p)$
$\mathcal{M} \models pre(n_1, n_2)$ for each $pre(n_1, n_2) \in S_{S_0}$,
$\mathcal{M} \models NumTkns(k, S_0) = k$ where $NumTkns(k, S_0) = k \in S_{S_0}$,
which assures that $\mathcal{M}$ is an isomorphism to some $\mathbf{P}$ in $\mathbf{P}^{ini}$. □

To this end, we are able to obtain the following satisfiability theorem for SCOPE.

Theorem 6 *SCOPE is satisfiable.*

Proof. Since $\mathcal{S}_{una\_S_0}$ is satisfiable by any $\mathcal{M}$ corresponding to some $\mathbf{P} \in \mathbf{P}^{ini}$, $\mathcal{S}$ is also satisfiable, as by Pirri and Reiter's Relative Satisfiability Theorem ([9], [10]), $\mathcal{M}$ can be extended to some $\mathcal{M}'$, which is a model of $\mathcal{S}$. $\square$

5. Applications of SCOPE

5.1. Axiomatization of SCOPE Subclasses

In this section, we provide an axiomatization (called $\mathcal{S}_{ps}$) to the subclasses and properties of Petri nets mentioned in Section 2. With $\mathcal{S}_{ps}$ and $\mathcal{S}$, two important tasks specified as follows, for the Petri nets Ontology designers, developers, and users can be performed: (1) to assure that the property say $\alpha \in \mathcal{S}_{ps}$ holds in the ontology to be developed; or (2) to check whether $\alpha \in \mathcal{S}_{ps}$ can be entailed from the existing ontology; that is, we can (1) check $\mathcal{S} \cup \alpha$ is satisfiable; or (2) check $\mathcal{S} \models \alpha$.

The specifications might use the following two abbreviations: [4]

$$enabled(t, s) \overset{def}{=} pre(p, t) \supset NumTkns(p, s) \geq 1$$

$$exec(s) \overset{def}{=} (\forall a, s).do(a, s) \sqsubseteq s \supset Poss(a, s).$$

Dynamical properties of Petri nets are defined as follows.

Cycle

$$(\exists s).\ exec(s) \wedge s \neq S_0 \wedge NumTkns(p, s) = NumTkns(p, S_0)$$

Reachability Given M_n, certain specified Marking on places, we have

$$(\exists s).\ exec(s) \wedge NumTkns(p_0, s) = n_0 \wedge \ldots NumTkns(p_i, s) = n_i,$$

where n_i is the specified number of tokens at the place p_i in the Marking M_n.

K-Boundedness

$$exec(s) \supset NumTkns(p, s) \leq k$$

Liveness

$$exec(s) \supset (\exists s_1).\ s \sqsubset s_1 \wedge exec(s_1)$$

Reversibility

$$exec(s) \supset (\exists s_1).\ s \sqsubset s_1 \wedge exec(s_1) \wedge NumTkns(p, s_1) = NumTkns(p, S_0),$$

where S_0 is the home situation corresponding the home marking of Petri nets.

[4] The second one is introduced as Equation 4.5 in [10].

Coverability (s_1 is covered by s_2)

$$(\exists s_1, s_2).\, exec(s_1) \wedge exec(s_2) \wedge NumTkns(p, s_1) \leq NumTkns(p, s_2)$$

Persistence

$$\neg(\exists s, t_1, t_2).\, exec(s) \wedge enabled(t_1, s) \wedge enabled(t_2, s) \wedge \neg enabled(t_1, do(fire(t_1), s))$$

Restricted classes of Petri nets are defined as follows.

S-systems

$$pre(p_1, t) \wedge pre(p_2, t) \supset p_1 = p_2,\ post(p_1, t) \wedge post(p_2, t) \supset p_1 = p_2$$

T-systems

$$pre(t_1, p) \wedge pre(t_2, p) \supset t_1 = t_2,\ post(t_1, p) \wedge post(t_2, p) \supset t_1 = t_2$$

Conflict-Free

$$\exists(t_1, t_2)\, post(p, t_1) \wedge post(p, t_2) \wedge t_1 \neq t_2 \supset pre(t_3, p) \wedge post(p, t_3)$$

Free-Choice

$$pre(p, t) \supset (\neg \exists p_1\, pre(p_1, t) \wedge p_1 \neq p) \vee (\neg \exists t_1\, pre(p, t_1) \wedge t_1 \neq t)$$

5.2. Examples

We now demonstrate the use of SCOPE by giving two examples. They are adapted from Figure 17 (h) of [5], which is a live, reversible, but 1-bounded Petri net instance, and Figure 17 (c) of [5], which is a non-live, irreversible, and 1-bounded Petri net instance. Their pictorial representations are shown in Figure 1 as (A) and (B). Also, the principle of induction for executable situations (PIES) as a consequence of $\mathcal{S}$ (see also [10]) turns out to be helpful:

$$(\forall P).P(S_0) \wedge (\forall a, s)\big(P(s) \wedge exec(s) \wedge Poss(a, s) \supset P(do(a, s))\big) \supset (\forall s).exec(s) \supset P(s).$$

Example 1 $\mathcal{S}_1 = \mathcal{D}_f \cup \mathcal{S}_{ap} \cup \mathcal{S}_{ss} \cup \mathcal{S}_{una} \cup \mathcal{S}_{S_0}$ *where*

$$\mathcal{S}_{S_0} = \{pre(p, t) \equiv post(t, p), pre(T_1, P_1), pre(P_1, T_2), pre(T_2, P_2), pre(P_2, T_1),$$

$$NumTkns(P_1, S_0) = 1, NumTkns(P_2, S_0) = 0.\}$$

1-boundedness: $\mathcal{S}_1 \models exec(s) \supset NumTkns(p, s) \leq 1.$
Proof. We prove 1-boundedness by showing
$exec(s) \supset [(NumTkns(P_1, s) = 1 \wedge NumTkns(P_2, s) = 0) \vee (NumTkns(P_2, s) = 1 \wedge NumTkns(P_1, s) = 0)].$

Now define $\psi(s) \overset{def}{=} (NumTkns(P_1, s) = 1 \wedge NumTkns(P_2, s) = 0) \vee$ $(NumTkns(P_2, s) = 1 \wedge NumTkns(P_1, s) = 0)$,

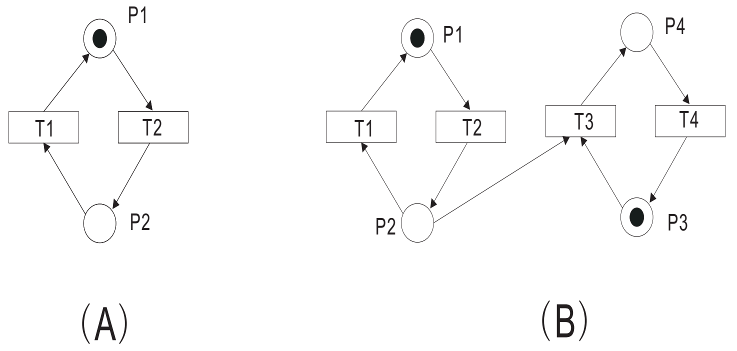

Figure 1. Two examples of Petri nets: (A) 1-bounded, live, and reversible; (B) 1-bounded, non-live, and irreversible.

thus $\psi(do(fire(t), s)) \overset{def}{=}$
$(NumTkns(P_1, do(fire(t), s)) = 1 \wedge NumTkns(P_2, do(fire(t), s)) = 0)$
$\vee(NumTkns(P_2, do(fire(t), s)) = 1 \wedge NumTkns(P_1, do(fire(t), s)) = 0)$.

$\psi(S_0)$ holds trivially, as $(NumTkns(P_1, S_0) = 1 \wedge NumTkns(P_2, S_1) = 0)$.

Now assume $\psi(s) \wedge exec(s) \wedge Poss(a, s)$ holds, we have two cases to consider

1. $(NumTkns(P_1, s) = 1 \wedge NumTkns(P_2, s) = 0)$
2. $(NumTkns(P_2, s) = 1 \wedge NumTkns(P_1, s) = 0)$

By using the SSA, for (1)
$(NumTkns(P_2, do(fire(t), s)) = 1 \wedge NumTkns(P_1, do(fire(t), s)) = 0)$
will be obtained, for (2)
$(NumTkns(P_2, do(fire(t), s)) = 0 \wedge NumTkns(P_1, do(fire(t), s)) = 1)$
will hold, obviously, (1) and (2) lead to the holdness of $\psi(do(fire(t), s))$; that is,
$(\forall a, s)[\psi(s) \wedge exec(s) \wedge Poss(a, s) \supset \psi(do(a, s))$.

By PIES, we have $(\forall s).exec(s) \supset \psi(s)$.
And finally, $exec(s) \supset NumTkns(p, s) \leq 1$. $\square$

Liveness We also use the two cases. To prove $exec(s) \supset \exists s_1\ s \sqsubset s_1 \wedge exec(s_1)$ by proving

1. $exec(s) \wedge (NumTkns(P_1, s) = 1 \wedge NumTkns(P_2, s) = 0) \supset \exists s_1\ s \sqsubset s_1 \wedge exec(s_1)$;
2. $exec(s) \wedge (NumTkns(P_1, s) = 0 \wedge NumTkns(P_2, s) = 1) \supset \exists s_1\ s \sqsubset s_1 \wedge exec(s_1)$.

For (1), we will show that $do(fire(T_2), s)$ is such kind of s_1; similarly, for (2), we have $do(fire(T_1), s)$.

Reversibility We just show that any situation is the same as S_0 or its subsequent situation is S_0.

Example 2 $\mathcal{S}_2 = \mathcal{D}_f \cup \mathcal{S}_{ap} \cup \mathcal{S}_{ss} \cup \mathcal{S}_{una} \cup \mathcal{S}_{S_0}$ *where* $\mathcal{S}_{S_0} = \{pre(p,t) \equiv post(t,p), pre(T_1,P_1), pre(P_1,T_2), pre(T_2,P_2), pre(P_2,T_1),$
$pre(T_4,P_3), pre(P_3,T_3), pre(T_3,P_4), pre(P_4,T_4), pre(P_2,T_3),$
$NumTkns(P_1,S_0) = 1, NumTkns(P_3,S_0) = 1,$
$NumTkns(P_2,S_0) = 0, NumTkns(P_4,S_0) = 0.\}$

Observe, and also it is easy to see that the theory entails, that

- T_4 is possible to execute only if T_3 is executed;
- T_1 and T_2 execute in turn from initial situation S_0 before T_3 can execute;
- the number of tokens at both P_1 and P_2 will be zero after the execution of T_3;
- no successor situation after the execution of T_4 is executable.

Thus the non-liveness of $\mathcal{S}_{example}$ is entailed:
$\mathcal{S}_{example} \models \neg(exec(s) \supset \exists s_1 s \sqsubset s_1 \wedge exec(s_1))$,
as, for example, at the executable situation $do(fire(T_3), do(fire(T_2), S0))$ the transition T_1 is no longer live. Meanwhile, the irreversibility is implied from this non-liveness, for this example in particular.

In order to show the 1-boundedness, we can show that, after T_3 is executed, T_4 must be executed and deadlock is reached, whereas all places are bounded before T_3 is executed. More specifically, we can use PIES to show that at any of these executable situations, we have either $NumTkns(P_1,S_0) = 1 \wedge NumTkns(P_2,S_0) = 0$,
or $NumTkns(P_1,S_0) = 0 \wedge NumTkns(P_2,S_0) = 1$.

Thus the example is also 1-bounded:
$\mathcal{S}_{example} \models exec(s) \supset NumTkns(p,s) \leq 1$.

6. Summary

In this paper, we provide a formal (mostly in first-order logic) ontology called SCOPE for Petri nets. The design of SCOPE makes the full use of the strong correspondence between the situation s in situation calculus, which is a sequence of actions starting from the initial state S_0, and the marking M in Petri nets, which results from a sequence of transition firings starting from the initial marking M_0. Consequentially, SCOPE inherits completely the simplicity in formalism of Petri nets and their powerful expressive capability.

We define SCOPE as a Basic Action Theory of situation calculus such that we can take advantage of the Relative Satisfiability Theorem for Basic Action Theories and show in an easy and straightforward manner that SCOPE is satisfiable. This efforts endorses the qualification of SCOPE, to be an operational language describing Web service in semantic Web, or other use cases where unambiguous interpretations of operational semantics are strictly required.

Aside from the satisfiability of SCOPE, this paper introduces a combinatorial structure and uses it to fully characterize some core subset axioms ($\mathcal{S}_{una\_S_0}$) of SCOPE by providing the axiomatizability theorem and the satisfiability theorem for $\mathcal{S}_{una\_S_0}$. We will consider characterizing SCOPE completely in a future paper. Note that we often call $\mathcal{S}_{una\_S_0}$ as a *domain theory*, since, different from other sets in the ontology, it is domain dependent.

It is somewhat surprising how much reasoning tasks this small ontology can support. While it is the case that this benefit is largely empowered by applying second-order based foundational axioms, in reality, any concrete reasoning task can always use a replacement of a set of first-order axioms.

Major related work includes [7], and [3]. In [3], the semantics are first described by UML classes models, and then redirected into Protege, which is a logic-based ontology development tool, and finally presented in (Resource Description Framework) RDF and Web Ontology Language (OWL) formalisms. Given that the semantics of UML classes models have yet to be formally specified, model-theoretic results on this complicated semantic mapping process are not given in [3] and are intrinsically difficult to obtain indeed. Similarly, the Generic Process Model (GPM) semantic for Petri net, proposed in [7], lacks the arguments of the unique interpretations of GPM.

Future work on and beyond SCOPE can be divided into four categories as follows. One, extend SCOPE to include more features of high-level Petri nets; two, formally characterize several subclasses of SCOPE theories such that reasoning with specific queries are tractable; three, implement a stand-alone SCOPE reasoning system, preferably in the logic programming language Prolog; four, investigate SCOPE in concurrent or distributed systems, incorporating earlier work such as [8].

Acknowledgements

Technical guidance from Professor Michael Gruninger and the financial support of the NIST grant 60NANB7D6143 to this work are deeply appreciated. I am grateful to the anonymous reviewers for their detailed comments and suggestions.

References

[1] R. Brachman and H. Levesque, Knowledge representation and reasoning, Morgan Kaufmann, 2004.

[2] F. Lin, Situation Calculus, In F. Harmelen, V. Lifschitz, and B. Porter, editors, Handbook of Knowledge Representation, (2008), 649-669, Elsevier.

[3] D. Gasevic and V. Devedzic J. Davies, Reusing Petri nets through the semantic Web, *ESWS, LNCS 3053*,(2004), 284-298.

[4] S. Narayanan and S. McIlraith, Analysis and simulation of Web services, *Computer networks*, **42**, (2003), 675-693.

[5] T. Murata, Petri nets: properties, analysis and applications, *Proceedings of the IEEE*, **77(4)**, (1989), 541-580.

[6] J. McCarthy and P. Hayes, Some philosophical problems from the standpoint of artificial intelligence, In B. Meltzer and D. Michie, editors, *Machine Intelligence* (1969), 463-502, Edinburgh University Press, Edinburgh, Scotland.

[7] P. Soffer, M. Kaner, and Y. Wand, Assigning ontology-based semantics to process models: the case of Petri nets, *CAiSE, LNCS5074*, (2008), 16-31.

[8] M. Nielsen and V. Sassone, Petri Nets and Other Models of Concurrency. In: Lectures on Petri Nets I: Basic Models. (1998), 587-642.

[9] F. Pirri and R. Reiter, Some contributions to the metatheory of the situation calculus, *J. ACM*, **46(3)** (1999), 325-361.

[10] R. Reiter, Knowledge in Actioin: Logical foundations for describing and implementing dynamical systems, The MIT Press, 2001.