UNIVERSITÀ DEGLI STUDI DI UDINE

Scuola Superiore *Di Toppo Wassermann*

4^{th} year thesis

Well-Structured Transition Systems

SUPERVISOR
Prof. Gabriele Puppis
TUTOR
Prof. Alberto Policriti

STUDENT
Cristian Curaba

Academic Year 2022/2023

Contents

Introduction

The concept of a well-structured transition system (WSTS) arose thirty years ago ([Fin90],[Ott87]) when such systems were initially called structured transition systems and were shown to have decidable termination and boundedness problems.

WSTS were developed for the purpose of capturing properties common to a wide range of formal models used in model-checking, system verification and concurrent programming. The coverability (Theorem 3.1) for such systems was shown decidable in 1996 ([Abd+96]), thus generalizing the decidability of coverability for a wide range of formal models (e.g. Petri nets and extensions or Context Free Grammars).

The usefulness of the WSTS stemmed from its clear abstract treatment of the properties responsible for the decidability of coverability, termination and boundedness. Because of this, an intensive development of the theory of WSTS has been followed. At its core, a WSTS is simply a set X (of states, possibly infinite) with a transition relation $\rightarrow \subseteq X \times X$. The set X is quasi-ordered by $\leq$, and $\rightarrow$ fulfils one of the various possible monotonicities, i.e. compatibilities with $\leq$. The quasi-ordering of X is further assumed to be "well", i.e. well-founded (Definition 2.2).

Over the years, a number of strengthenings and weakenings of the notion of monotonicity (of $\rightarrow$ w.r.t. $\leq$) were introduced, with the goal of allowing WSTS to capture even more models.

In this document, we will provide a formal introduction to well-structured transition systems starting from the fundamentals. This introduction will primarily follow the statements outlined in the article by Finkle ([Fin90]), incorporating detailed explanations and illustrative examples, such as the broadcast protocol. Section 3 provides a concise overview of two classical approaches for proving decidability results employed in formal methods, named the set-saturation method and the tree-saturation method, within the context of Well-Structured Transition Systems (WSTS). Additionally, we introduce Petri nets, a well-established formal system for modelling concurrent transitions, and we show how classical problems can be proven by formalizing them within the framework of WSTS.

1 Well-quasi orderings

Recall that a quasi-ordering (a qo) is any reflexive and transitive relation $\leq$. We let $x < y$ denote $x \leq y$ & $x \neq y$. A partial ordering (a po) is an antisymmetric qo.

Definition 1.1. *A* well-quasi-ordering *(a wqo) is any quasi-ordering $\leq$ (over some set X) such that, for any infinite sequence $x_0, x_1, x_2, \dots$ in X, there exists indexes $i < j$ with $x_i \leq x_j$.*

Hence a wqo is well-founded, i.e. it admits no infinite strictly decreasing sequence $x_0 > x_1 > x_2 > \cdots$

Osservation 1. *We notice that any quasi-ordering in a finite set X is also a well-quasi-ordering: any infinite sequence $x_0, x_1, \dots$ must contain two indexes $i < j$ such that $x_i = x_j$, hence $x_i \leq x_j$.*

Lemma 1.1 (Erdös & Rado)**.** *Assume $\leq$ is a wqo. Then any infinite sequence contains an infinite increasing subsequence: $x_{i_0} \leq x_{i_1} \leq x_{i_2} \dots$ (with $i_0 < i_1 < i_2 \dots$.*

Proof. Consider an infinite sequence and the set $M = \{i \in \mathbb{N} \mid \forall j > i, x_i \nleq x_j\}$. M cannot be infinite, otherwise, it would lead to an infinite subsequence contradicting the wqo hypothesis. Let i_0 be any index greater than $\max M$, then i_0 can start an infinite increasing subsequence. □

The existence of such infinite increasing subsequences is sometimes taken as a definition for well-quasi-ordering, since leads to an equivalent definition (the other implication is obvious).

Example 1 (Power set)**.** *Given a quasi-ordering $\leq$ for a set X, we can define $\leq^+$ on X's power set $\mathcal{P}(X)$ as follows:*

$$A \leq^+ B \iff \forall a \in A \, \exists b \in B \text{ s.t. } a \leq b.$$

If $\leq$ is not well-founded, then neither $\leq^+$ is well-founded: we can consider $(\mathbb{Q}, \leq)$ with the natural order. By taking the sequences $x_n = \frac{1}{n}$ in $\mathbb{Q}$ and $y_n = \{\frac{1}{n}\}$ in $\mathbb{Q}$ we contradict the well-foundness propriety.

Whenever $\leq$ is well-founded, then also $\leq^+$ is well-founded. Let's prove it by contradiction: suppose there exists a sequence $A_1, A_2, \dots$ in X s.t. no indexes $i < j$ satisfy $A_i \leq^+ A_j$. Let's consider $a_1 \in A_1, a_2 \in A_2, \dots$ s.t. a_i is an element of A_i that for each $a_{i+1} \in A_{i+1}$ $a_{i+1} < a_i$, then $a_1 > a_2 > a_3 > \dots$ contradicts well-foundness propriety of $(X, \leq)$.

Example 2 (($\mathbb{N}^k, \leq_x$) is wqo)**.** *Let $k \in \mathbb{N}$, $\mathbf{a} = (a_1, \dots, a_k)$ and $\mathbf{b} = (b_1, \dots, b_k)$ in $\mathbb{N}^k$. We can define $\mathbf{a} \leq_x \mathbf{b} \iff a_1 \leq b_1 \wedge a_2 \leq b_2 \wedge \cdots \wedge a_k \leq b_k$. We can show that $(\mathbb{N}^k, \leq_x)$ is a wqo. Consider an infinite sequence $\mathbf{a_1}, \mathbf{a_2}, \dots$ over $\mathbb{N}^k$ and write $\mathbf{a_i} = (a_{i,1}, \dots, a_{i,k})$. We can extract an infinite subsequence $\mathbf{a_{i_1}}, \mathbf{a_{i_2}}, \dots$ that is increasing over the first components, i.e., with $a_{i_1,1} \leq a_{i_2,1} \leq a_{i_3,1} \leq$ since $(\mathbb{N}, \leq)$ is a wqo. From this infinite subsequence, we can further extract an infinite subsequence that is also increasing on the second component (again, using that N is wqo). We can iterate the same until the last k component is reached.*

Definition 1.2. *Given $\leq$ a quasi-ordering, an* upward-closed set *is any set $I \subseteq X$ such that $y \geq x$ and $x \in I$ entail $y \in I$.*

Notation. *To any $x \in X$ we write $\uparrow x \overset{def}{=} \{y \mid y \geq x\}$.*

To any $K \subseteq X$ we write $\uparrow K \overset{def}{=} \{x \in X \mid \exists k \in K\ x \geq k\}$.

Definition 1.3. *A* basis of an upward-closed I *is a set I^b such that*

$$I = \bigcup_{x \in I^b} \uparrow x.$$

Lemma 1.2. *If $\leq$ is a wqo, then any upward-closed set I has a finite basis.*

Proof. The set of minimal elements of I is a basis because $\leq$ is well-founded. It must be finite since a wqo does not admit an infinite strictly decreasing sequence. □

The following lemma is fundamental for ensuring the termination of set-saturation methods which will provide a demonstration of the decidability of the *covering problem* for WSTS (with mild assumptions). We will see it in Section 3.1.

Lemma 1.3. *If $\leq$ is a wqo, any infinite increasing sequence $I_0 \subseteq I_1 \subseteq I_2 \dots$ of upward-closed sets eventually stabilizes, i.e. there is a $k \in \mathbb{N}$ such that $I_k = I_{k+1} = I_{k+2} = \dots$*

Proof. Assume we have a counter-example. We can extract an infinite subsequence where inclusion is strict: $I_{n_0} \subsetneq I_{n_1} \subsetneq I_{n_2} \subsetneq \dots$. Now, for any $i > 0$, we can pick some $x_i \in I_{n_i} \setminus I_{n_{i-1}}$. For the definition of well-quasi-ordering, the sequence of x_i contains a pair $i < j$ such that $x_i \leq x_j$. Because x_i belongs to an upward-closed set I_{n_i} we have $x_j \in I_{n_i}$, contradicting $x_j \notin I_{n_{i-1}}$. □

2 Well Structured Transition Systems

Definition 2.1. *A transition system (TS) is a structure* $\mathcal{S} = \langle S, \rightarrow, \ldots \rangle$ *where* $S = \{s, t, \ldots\}$[1] *is a set of states, and* $\rightarrow \subseteq S \times S$ *is any set of transitions.*

Notation. *In the following list, we establish some intuitive notations and definitions for transition systems.*

- *We write Succ(s) (resp. Pred(s)) for the set* $\{s' \in S \mid s \rightarrow s'\}$ *of immediate successors of s (resp.* $\{s' \in S \mid s' \rightarrow s\}$ *the immediate predecessors).*
- *A state with no successor is a* terminal state.
- *A* computation *is a maximal sequence* $s_0 \rightarrow s_1 \rightarrow s_2 \cdots$ *of transitions.*
- *We write* $\xrightarrow{n}$ *(resp.* $\xrightarrow{+}, \xrightarrow{=}, \xrightarrow{x}$ *) for the n-step iteration of the transition relation* $\rightarrow$ *(resp. for its transitive closure, for its reflexive closure, for its reflexive and transitive closure). Hence* $\xrightarrow{1}$ *is* $\rightarrow$*.*
- *We use similar notation for Succ() and Pred(), so that for* $\alpha \in \{+, =, *, 0, 1, 2, \ldots\}, \mathrm{Succ}^{\alpha}(s)$ *is* $\left\{s' \mid s \xrightarrow{\alpha} s'\right\}$.

The transition system $\mathcal{S}$ is finitely branching if all Succ(s) are finite. We restrict our attention to *finitely branching TS's.*

Definition 2.2. *A well-structured transition system (WSTS) is a TS* $\mathcal{S} = \langle S, \rightarrow, \leq \rangle$ *equipped with a qo* $\leq \subseteq S \times S$ *between states such that the two following conditions hold:*

- *well-quasi-ordering:* $\leq$ *is a wqo*
- *compatibility (or monotonicity):* $\leq$ *is (upward) compatible with* $\rightarrow$*, i.e. for all* $s_1 \leq t_1$ *and transition* $s_1 \rightarrow s_2$*, there exists a sequence* $t_1 \rightarrow t_2$ *such that* $s_2 \leq t_2$*.*

We can consider a slightly strengthened definition by requiring *strict compatibility.*

Definition 2.3. *A well-structured transition system (WSTS) S has* strict compatibility *if for all* $s_1 < t_1$ *and transition* $s_1 \rightarrow s_2$ *exists a sequence* $t_1 \xrightarrow{*} t_2$ *such that* $t_1 < t_2$*.*

[1] TS's may have additional structure like initial states, labels for transitions, durations, causal independence relations, etc.

Strict compatibility means that from strictly larger states it is possible to reach strictly larger states. Several families of formal models of processes give rise to WSTS's in a natural way, e.g. Petri nets when inclusion between markings is used as the well-ordering (look at Section 4 for details).

Transitive and stuttering compatibility To apply decidability results to the largest set of formal methods, we need to grasp the weakest extension of the "standard" compatibility (Definition 2.2) to prove a decidability result. For this purpose, we will consider slightly different extensions of compatibilities.

Definition 2.4. *Let $\mathcal{S}$ be a WSTS.*

- $\mathcal{S}$ *has* strong compatibility *(also called "1-1 compatibility") if for all $s_1 \leq t_1$ and transition $s_1 \rightarrow s_2$, there exists a transition $t_1 \rightarrow t_2$ with $s_2 \leq t_2$;*
- $\mathcal{S}$ *has* transitive compatibility *if for all $s_1 \leq t_1$ and transition $s_1 \rightarrow s_2$, there exists a non-empty sequence $t_1 \rightarrow t_2 \rightarrow \cdots \rightarrow t_n$ with $s_2 \leq t_n$.*
- $\mathcal{S}$ *has* stuttering compatibility *if for all $s_1 \leq t_1$ and transition $s_1 \rightarrow s_2$, there exists a non-empty sequence $t_1 \rightarrow t_2 \rightarrow \cdots \rightarrow t_n$ with $s_2 \leq t_n$ and $s_1 \leq t_i$ for all $i < n$.*
- $\mathcal{S}$ *has* reflexive compatibility *if for all $s_1 \leq t_1$ and transition $s_1 \rightarrow s_2$, either $s_2 \leq t_1$ or there exists a transition $t_1 \rightarrow t_2$ with $s_2 \leq t_2$.*

These proprieties can be strengthened by considering *strict compatibility* as done in Definition 2.3.

Osservation 2. *When $\mathcal{S} = \langle S, \rightarrow, \leq \rangle$ is a WSTS, then $\mathcal{S}^* \overset{def}{=} \langle S, \overset{*}{\rightarrow}, \leq \rangle$ has strong, "1-1", compatibility, but it is not necessarily a WSTS. $\mathcal{S}^*$ is in general (when cardinality of S is greater or equal to $\aleph_0$) not finitely branching. Worse, when effectiveness issues are taken into account, $\mathcal{S}^*$ needs not have effective Succ or pred-basis even when $\mathcal{S}$ has.*

We now illustrate three important examples of formal systems which can be formalized in the context of WSTS: lossy channel systems, broadcast protocols and context-free grammars.

Lossy Channel Systems [Abd+04] We consider system models consisting of asynchronous parallel compositions of finite-state machines that communicate through sending and receiving messages via a finite set of unbounded lossy FIFO channels (in the sense that they can nondeterministically lose messages).

Definition 2.5. *A Lossy Channel System (LCS) $\mathcal{L}$ is a tuple $(S, s_{init}, C, M, \delta)$*[2]*, where*

- *S is a finite set of (control) states. The control states of a system with n finite-state machines are formed as the Cartesian product $S = S_1 \times \cdots \times S_n$ of the control states of each finite-state machine;*
- *$s_{init} \in S$ is an initial state. The initial state of a system with n finite state machines is a tuple $\langle s_{init\,1}, \ldots, s_{init\,n} \rangle$ of initial states of the components;*
- *C is a finite set of channels;*
- *M is a finite set of messages;*
- *δ is a finite set of transitions, each of which is of the form (s_1, Op, s_2), where s_1 and s_2 are states, and Op is a mapping from C to (channel) operations. An operation is either a send operation $a!$, a receive operation $a?$, or an empty operation nop, where $a \in M$.*

Notation. *Let's set up some frequently used notations.*

- *For $x, y \in M^*$, we let $x \bullet y$ denote the concatenation of x and y.*
- *We use x^n to denote the concatenation of n copies of x.*
- *The empty string is denoted by ε.*
- *We use $\preceq$ to denote the subsequence relation on M^*, i.e., $x \preceq y$ denotes that x is a (not necessarily contiguous) substring of y.*

Example 3 (The Alternating Bit Protocol)**.** *In this example, we model the well-known Alternating Bit Protocol as a lossy channel system. The alternating bit protocol contains a Sender and a Receiver that communicate over two FIFO channels c_M (used to transmit messages from the Sender to the Receiver) and c_A (used to transmit acknowledgements from the Receiver to*

[2]Sometimes a finite set of transition labels can be added: it's practically useful to describe protocols but uncomfortable for formal proves.

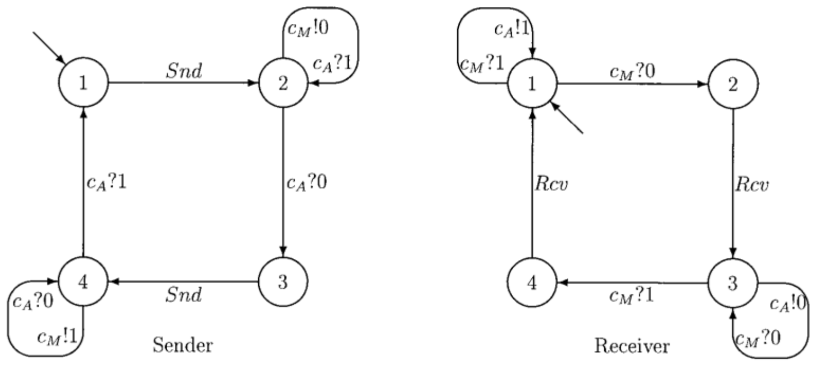

Figure 1: The Sender and the Receiver of the Alternating Bit Protocol.

the Sender). Both channels are faulty in the sense that they can lose but not reorder messages. The purpose of the protocol is to transmit messages from the Sender to the Receiver in the correct order, in spite of the fact that the channels can lose messages. Corruption of messages can also be taken into account by modelling them as loss (some mechanism will detect and discard a corrupted message).

The operation of the protocol is the following: the Sender reads a pending message to be sent to the Receiver. It adds a sequence number to the message, sends it over the channel c_M to the Receiver and awaits an acknowledgement from the Receiver with the same sequence number. If it arrives, the procedure is repeated with the next pending message but with sequence numbers inverted. If no acknowledgement arrives within some time period the Sender re-transmits the message. Re-transmissions are repeated until a corresponding acknowledgement arrives. The Receiver receives messages with accompanying sequence numbers from the channel c_M. When the message has the expected sequence number, the message is delivered, and the Receiver looks for a message with the inverted sequence number. Messages with unexpected sequence numbers are discarded. The Receiver sends acknowledgments to the Sender over the channel c_A. An acknowledgement contains the sequence number of the last received message. In Fig. 1 the Sender and the Receiver are represented by labelled transition systems. In our model we have omitted the actual messages; i.e., only sequence numbers are transmitted over the channels. The finite state control part of the lossy channel system is obtained as the combination of these two transition systems. The protocol operates on the two channels c_M and c_A. This means that the model of the Alternating Bit Protocol is the lossy channel system $\langle S, s_0, A, C, M, \delta \rangle$, where

- *S is the set of pairs of the form $\langle i, j \rangle$, where $1 \leq i, j \leq 4$, s_0 is the state*

$\langle 1, 1 \rangle$;

- *A is the set $\{Snd, Rcv\}$, where Snd represents the sending of a message by the environment to the protocol, and Rcv represents the reception of a message by the environment from the protocol;*
- *C is the set $\{c_M, c_A\}$;*
- *M is the set $\{0, 1\}$, i.e., messages consist of only a sequence number;*
- *δ consists of the tuples of the form $\langle\langle s_1, r_1\rangle, op, \langle s_2, r_2\rangle\rangle$, where either $r_1 = r_2$ and $\langle s_1, op, s_2\rangle$ is a transition in the Sender component or $s_1 = s_2$ and $\langle r_1, op, r_2\rangle$ is a transition in the Receiver component.*

Let's show that lossy channel systems are WSTS.

Theorem 2.1. *Let $\preceq$ be the subword ordering. Lossy channel systems are WSTS with stuttering compatibility.*

Proof. Let $\mathcal{L} = (S, s_{\text{init}}, C, M, \delta)$ and let's consider the transition system $\mathcal{S}_{\mathcal{L}}$ where a configuration is any $k = \langle s, w_1, \ldots, w_n\rangle$ where $s \in S$ is the control state and $w_i \in M^*$ is the current content of the channel $c_i \in C$. Let's consider the transition system $(\mathcal{S}_{\mathcal{L}}, \rightarrow)$ where $\rightarrow \subseteq \mathcal{S}_{\mathcal{L}} \times \mathcal{S}_{\mathcal{L}}$ is given by $\delta \subseteq S \times Op \times S$ as follows:

$$\begin{aligned}&\forall k = \langle s, w_1, \ldots, w_n\rangle, \forall k' = \langle s', w'_1, \ldots, w'_n\rangle \quad k \rightarrow k' \iff\\ &\exists (s, \text{op}, s') \in \delta,\ \text{op maps } (w_1, \ldots, w_n) \text{ into } (w'_1, \ldots, w'_n).\end{aligned}$$

Let's derive an order for $\mathcal{S}_{\mathcal{L}}$ from the subword ordering as follows:

$$\langle s, w_1, \ldots, w_n\rangle \preceq \langle s', w'_1, \ldots, w'_n\rangle \iff \begin{cases} s = s' \text{ and} \\ w_i \preceq w'_i \text{ for } i = 1, \ldots, n. \end{cases}$$

$(\mathcal{S}_{\mathcal{L}}, \preceq)$ is a wqo. Let's prove that $((\mathcal{S}_{\mathcal{L}}, \preceq), \rightarrow)$ is a WSTS with stuttering compatibility. Let $k_1 \preceq j_1$ and $k_1 \rightarrow k_2$. To find a non-empty sequence $j_1 \rightarrow j_2 \rightarrow \cdots \rightarrow j_n$ with $k_2 \leq j_n$ and $k_1 \leq j_i$ for all $i < n$. We can handle the messages on each channel by considering the set of transitions $\delta \colon C \rightarrow Op^3$. □

[3]We notice that the stuttering propriety can be satisfied even if channels are lossy.

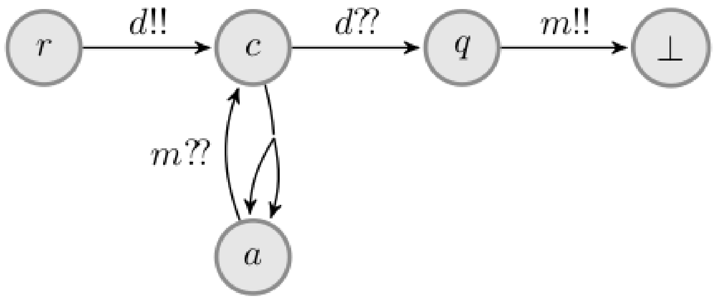

Figure 2: A broadcast protocol.

Broadcast Protocols [SS14] As a concrete illustration of the principles behind WSTS, let us consider a broadcast protocol in distributed systems. In such protocols, an unbounded number of identical finite-state processes running concurrently, can spawn new processes and communicate either via exchanging a message between two processes (named *rendez-vous* message) or via broadcast.

Definition 2.6. *A* broadcast protocol *is defined as a triple $B = (Q, M, R)$ where Q is a finite set of locations, M a finite set of messages, and R is a set of rules, that is, tuples (q, op, q') in $Q \times Op \times Q$, each describing an operation op available in the location q and leading to a new location q', where op can be:*

- *a sending* rendez-vous *message $m \in M$ (denoted $m!$);*
- *a receiving operation of a* rendez-vous *message $m \in M$ (denoted $m?$);*
- *a sending broadcast message $m \in M$ (denoted $m!!$);*
- *a receiving operation of a broadcast message $m \in M$ (denoted $m??$).*
- *a spawning of a new process that op will start executing from location p (denoted $sp(p)$).*

As usual, we write $q \xrightarrow{op} q'$ if (q, op, q') is in R.

Figure 2 displays a toy example where $Q = \{r, c, a, q, \perp\}$ and $M = \{d, m\}$: processes in location c can spawn new "active" processes in location a, while also moving to location a (a rule depicted as a double arrow in Figure 2). These active processes are flushed upon receiving a broadcast of either m (emitted by a process in location q) or d (emitted by a process in location r); location $\perp$ is a sink location modelling process destruction.

The operational semantics of a broadcast protocol are expressed as a transition system $\mathcal{S}_B = (S, \to)$, where states, here called configurations, are (finite) multisets of locations in Q, hence $S = \mathbb{N}^Q$. Informantics for a configuration s in $\mathbb{N}^Q$ is to record for each location q in Q the number of processes $s(q)$ currently in this location.

Notation. *A state $s = \{q_1^{n_1}, q_2^{n_2}, \dots\} \in \mathbb{N}^Q$ describes a multiset with $n_1 \in \mathbb{N}$ duplicates of $q_1 \in Q$, $n_2 \in \mathbb{N}$ duplicates of $q_2 \in Q$ and so on. We will denote the number of duplicates of an element in $q \in Q$ in a multiset $s \in \mathbb{N}^Q$ as $s(q)$, i.e., $n_1 = s(q_1), n_2 = s(q_2)$ and so on.*

A natural ordering for $\mathbb{N}^Q$ is the inclusion ordering defined by

$$s \subseteq s' \iff \forall q \in Q \; s(q) \leq s'(q).$$

For instance, $\{q^2, q'\} \subseteq \{q^3, q'\}$, but if $q \neq q'$, $\{q, q'^2\} \not\subseteq \{q^2, q'\}$. We further write $s = s_1 + s_2$ for the union of two multisets, in which case $s - s_1$ denotes $s2$.[4]

It remains to define how the operations of B update such a configuration through transitions $s \to s'$ of $\mathcal{S}_B$:

- rendez-vous step: if $q_1 \xrightarrow{m!}_B q_1'$ and $q_2 \xrightarrow{m?}_B q_2'$ for some $m \in M$, then $s + \{q_1, q_2\} \to s + \{q_1', q_2'\}$ for all $s \in \mathbb{N}^Q$,
- *spawn* step: if $q \xrightarrow{sp(p)}_B q'$, then $s + \{q\} \to s + \{q', p\}$ for all $s \in \mathbb{N}^Q$,
- *broadcast* step: if $q_0 \xrightarrow{m!!}_B q_0'$ and $q_i \xrightarrow{m??}_B q_i'$ for all $1 \leq i \leq k$ (and some $m \in M$), then $s + \{q_0, q_1, \dots, q_k\} \to s + \{q_0', q_1', \dots, q_k'\}$ for all $s \in \mathbb{N}^Q$ that do not contain a potential receiver for the broadcast, i.e., such that $s(q) = 0$ for all rules of the form $q \xrightarrow{m??}_B q'$.

With the protocol of Figure 2, the following steps are possible (with the spawned location or exchanged messages indicated on the arrows):

$$\{c^2, q, r\} \xrightarrow{a} \{a^2, c, q, r\} \xrightarrow{a} \{a^4, q, r\} \xrightarrow{m} \{c^4, r, \perp\} \xrightarrow{d} \{c, q^4, \perp\}.$$

We have just associated an ordered transition system $\mathcal{S}_B = (\mathbb{N}^Q, \to, \subseteq)$ with every broadcast protocol B and are now ready to prove the following fact.

Proposition 2.1. *Broadcast protocols are WSTS.*

[4] The $s_1 - s_2$ operation is well defined iff $s_2 \subseteq s_1$.

Proof. First, $(\mathbb{N}^Q, \subseteq)$ is a wqo: since Q is finite, this is just another instance of Example 2. There remains to check that $\mathcal{S}_B$ has compatibility. Formally, this is done by considering an arbitrary step $s_1 \to s_2$ (there are three cases) and an arbitrary pair $s_1 \subseteq t_1$. It is enough to assume that $t_1 = s_1 + \{q\}$, i.e., t_1 is just one location bigger that s_1, and to rely on transitivity. If $s_1 \to s_2$ is a *rendez-vous* step with $s_2 = s_1 - \{q_1, q_2\} + \{q_1', q_2'\}$, then $t_1 = s_1 + \{q\}$ also has a *rendez-vous* step $t_1 \to t_2 = t_1 - \{q_1, q_2\} + \{q_1', q_2'\}$ and one sees that $s2 \subseteq t2$ as required. If now if $s_1 \to s_2$ is a spawn step, similar reasoning proves that $s_1 + \{q\} \to s_2 + \{q\}$. Finally, when $s_1 = s + \{q_1, \dots\} \to s_2 = s + \{q_1', \dots\}$ is a broadcast step, one proves that $s_1 + \{q\} \to s_2 + \{q'\}$ when there is a rule $q \xrightarrow{m??}_B q'$, or when q is not a potential receiver and $q' = q$. □

The protocol depicted in Figure 1 always terminates, starting from any initial configuration. Indeed, consider any sequence of steps $s_0 \to s_1 \to \cdots \to s_i \to \cdots$, write each configuration under the form $s_i = \{a^{n_{a,i}}, c^{n_{c,i}}, q^{n_{q,i}}, r^{n_{r,i}}, \bot^{n_{\bot,i}}\}$, and compare any two s_i and s_j with $i < j$:

- either only spawn steps occur along the segment $s_i \to s_{i+1} \to \cdots \to s_j$, thus $n_{c,j} < n_{c,i}$;
- or at least one m has been broadcast but no d has been broadcast, thus $n_{q,j} < n_{q,i}$;
- or at least one d has been broadcast, and then $n_{r,j} < n_{r,i}$.

Thus in all cases, $s_i \not\subseteq s_j$, i.e. the sequence $s_0, s_1, \dots$ contradicts well-foundation. Since $(\mathbb{N}^Q, \subseteq)$ is a wqo there's no infinite run.

We will prove the termination propriety in the more general context of wqo (Theorem 3.2).

Context-Free Grammars Context-Free Grammars (abbreviated CFG) are a special kind of string rewrite system. Let's define it:

Definition 2.7. *A CFG is a tuple $G = \langle N_G, T_G, R_G \rangle$ where $N_G \cap T_G = \emptyset$ composing the alphabet $\Sigma_G = N_G \sqcup T_G$ and $R_G \subseteq N_G \times \Sigma_G^*$ is a finite set of production rules.*

Given G a CFG, N_G represents the non-terminal state, T_G represents the terminal states, rules R_G are stated as $Z \to w$, with $Z \in N_G$ and $w \in \Sigma_G^*$. Sometimes we can also consider a starter symbol $S \in N_G$. We can develop two kinds of approaches:

- we can focus on the language generated by G, $L(G_S) \coloneqq \{w \in \Sigma^* | S \xrightarrow{*} w\}$, non-deterministic pushdown automata recognize exactly the context-free languages;

- we can emphasize the rewrite steps, which give rise to a transition system S_G where states are elements of Σ_G^* (finite words) and R_G can be adapted to a transition $\rightarrow \subseteq \Sigma_G^* \times \Sigma_G^*$. Several natural orders can be defined between words.

Let's develop the second approach by defining some natural orders:

Definition 2.8. *Let $\Sigma = N_G \sqcup T_G$ for a CFG $G = \langle N_G, T_G, R_G \rangle$.*

- embedding*: a word $u \in \Sigma^*$ embeds into a word $v \in \Sigma^*$, denoted $u \preceq v$, iff u can be obtained by erasing letters from v;*

- left-factor*: a word $u \in \Sigma^*$ is a left-factor (or a prefix), written $u \leq_{\text{lf}} v$ iff $v = uw$ for some word $w \in \Sigma^*$.*

- Parikh*: $u \leq_P v$ iff a permutation of u is a subword of v;*

It's easy to show that $\preceq$ and $\leq_{lf}$ are partial orders (quasi-orders with antisymmetry) while $\leq_P$ is a quasi-order (transitivity and reflexivity).

Assuming a finite alphabet, $\preceq$ is a wqo while $\leq_{lf}$ is not.

Example 4. *Let's show with a simple example that $(\Sigma^*, \leq_{\text{lf}})$ is not well founded with $\Sigma = \{0, 1\}$. It is enough to consider the following sequence: $x_0 = 1, x_1 = 01, x_2 = 001, \ldots$. This is an infinite sequence but does not exist $i < j$ s.t. x_i is a prefix of x_j.*

Lemma 2.1. *Let $\leq_1$ and $\leq_2$ be two partial orders on the same set X, with $\leq_2$ larger than $\leq_1$ ($\leq_1 \subseteq \leq_2$). If $\leq_1$ is a wqo then also $\leq_2$ is a wqo.*

Proof. Let's consider any infinite sequence $x_0, x_1, \ldots$ in X. Then exists $i < j$ such that $x_i \leq_1 x_j$ which implies that $x_i \leq_2 x_j$. □

Since $\leq_P$ is larger than $\preceq$, then $\leq_P$ is a wqo.

Proposition 2.2. *For any context-free grammar G,*

1. *$\langle S_G, \preceq \rangle$ is a WSTS with strong strict compatibility;*

2. *$\langle S_G, \leq_P \rangle$ is a WSTS with strong strict compatibility;*

Proof. The idea is the same for both orderings. We just show the case with $\prec$.

Let any $s_1 \prec t_1$ and transition $s_1 \rightarrow s_2$, we want to find t_2 s.t. $t_1 \rightarrow t_2$ with $s_2 \prec t_2$; the idea is just to apply the exact sequence of rules (for obtaining $s_1 \rightarrow s_2$) on t_1 (this can be done since s_1 is embedded in t_1) we obtain t_2 with $t_1 \prec t_2$. □

3 Classical methods

3.1 Set-saturation methods

We speak of set-saturation methods when we have methods whose termination relies on Lemma 1.3. In this section, we illustrate the idea with the backward reachability method for the covering problem.

Assume $\mathcal{S} = \langle S, \rightarrow, \leq \rangle$ is a WSTS and $I \subseteq S$ is a set of states. Backward reachability analysis involves computing $\mathrm{Pred}^*(I)$ as the limit of the sequence $I_0 \subseteq I_1 \subseteq \ldots$ where $I_0 \overset{\text{def}}{=} I$ and $I_{n+1} \overset{\text{def}}{=} I_n \cup \mathrm{Pred}\,(I_n)$. The problem with such a general approach is that termination is not guaranteed.

For WSTS's, this can be solved when I is upward-closed:

Proposition 3.1. *If $I \subseteq S$ is an upward-closed set of states, then* $\mathrm{Pred}^*(I)$ *is upward-closed.*

Proof. Assume $s \in \mathrm{Pred}^*(I)$. Then $s \xrightarrow{*} t$ for some $t \in I$. If now $s' \geq s$ then upward-compatibility entails that $s' \xrightarrow{*} t'$ for some $t' \geq t$. Then $t' \in I$ and $s' \in \mathrm{Pred}^*(I)$. □

To compute $\mathrm{Pred}^*(I)$ we shall make a few decidability assumptions:

Definition 3.1. *A WSTS has an* effective pred-basis *if there exists an algorithm accepting any state $s \in S$ and returning a finite basis of* $\uparrow \mathrm{Pred}(\uparrow s)$.

Our definition is necessary for the generalized Theorem 3.1 we aim at.

Notation. *We will denote $pb(s)$ as a finite base of* $\uparrow \mathrm{Pred}(\uparrow s)$. *For $K \subseteq S$, we will denote with $pb(K)$ a finite base for* $\bigcup_{s \in K} \uparrow \mathrm{Pred}(\uparrow s)$.

Proposition 3.2 (Distributivity property of Pred and $\uparrow$ w.r.t union)**.** *Let $\{K_i\}_{i \leq n}$ with $K_i \subseteq S$. Then*

$$\uparrow \bigcup_{i \leq n} K_i = \bigcup_{i \leq n} \uparrow K_i; \tag{1}$$

$$\mathrm{Pred}(\bigcup_{i \leq n} K_i) = \bigcup_{i \leq n} \mathrm{Pred}(K_i). \tag{2}$$

Proof. Let's prove $\uparrow \bigcup_{i \leq n} K_i = \bigcup_{i \leq n} \uparrow K_i$.

($\subseteq$) Let $x \in \uparrow \bigcup_{i \leq n} K_i$. Then $\exists i \leq n$ and $\exists k_i \in K_i$ s.t $x \geq k_i$. Then $x \in \uparrow K_i$ and so $x \in \bigcup_{i \leq n} \uparrow K_i$.

($\supseteq$) Let $x \in \bigcup_{i \leq n} \uparrow K_i$. Then $\exists i \leq n$ $x \in \uparrow K_i$. Then $\exists k_i \in K_i$ s.t. $x \geq k_i$. Since $k_i \in \bigcup_{i \leq n} K_i$ we have $x \in \uparrow \bigcup_{i \leq n} K_i$.

Let's prove $\mathrm{Pred}(\bigcup_{i \leq n} K_i) = \bigcup_{i \leq n} \mathrm{Pred}(K_i)$.

($\subseteq$) Let $x \in \text{Pred}(\bigcup_{i \le n} K_i)$. Then $\exists i \le n$ and $k_i \in K_i$ s.t. $k_i \to x$. Thus, $x \in \text{Pred}(K_i)$ which implies $x \in \bigcup_{i \le n} \text{Pred}(K_i)$.

($\supseteq$) Let $x \in \bigcup_{i \le n} \text{Pred}(K_i)$. Then $\exists i \le n$ s.t. $x \in \text{Pred}(K_i)$. Then $\exists k_i \in K_i$ s.t. $k_i \to x$, then $k_i \in \bigcup_{i \le n} K_i$, thus $x \in \text{Pred}(\bigcup_{i \le n} K_i)$. $\square$

Now assume that $\mathcal{S}$ is a WSTS with effective pred-basis. Pick I^b a finite basis of I and define a sequence $K_0, K_1, \ldots$ of sets with $K_0 \overset{\text{def}}{=} I^b$, and $K_{n+1} \overset{\text{def}}{=} K_\text{n} \cup pb\,(K_n)$. Let m be the first index such that $\uparrow K_m = \uparrow K_{m+1}$. Such an m must exist by Lemma 1.3.

Lemma 3.1. *$\uparrow K_m = \uparrow \bigcup_{i \in \mathbb{N}} K_i$.*

Proof. For the definition of *pb* and the distributivity property of Pred and $\uparrow$ w.r.t. union, we have:

$$\uparrow Y = \uparrow Y' \text{ implies } \uparrow pb(Y) = \uparrow pb\,(Y')\,.$$

$\square$

Lemma 3.2. *$\uparrow \bigcup_i K_i = \text{Pred}^*(I)$.*

Proof. Use induction over n and show that

$$K_n \subseteq \uparrow K_n \subseteq \text{Pred}^*(I) = \uparrow \text{Pred}^*(I)$$

On the other hand, the definition of *pb* entails $\uparrow \text{Pred}^n(I) \subseteq \uparrow K_n$, so that

$$\text{Pred}^*(I) \subseteq \bigcup_{i \in \mathbb{N}} \uparrow K_i \subseteq \uparrow \bigcup_{i \in \mathbb{N}} K_i \subseteq \uparrow \text{Pred}^*(I)$$

$\square$

Proposition 3.3. *If $\mathcal{S}$ is a WSTS with effective pred-basis and decidable $\le$, then it is possible to compute a finite basis of Pred$^*(I)$ for any upward-closed $I = \bigcup_{x \in I^b} \uparrow x$, given I^b.*

Proof. The sequence $K_0, K_1, \ldots$ can be constructed effectively (each K_n is finite and *pb* is effective). The index m can be computed because the computability of $\le$ entails the decidability of the predicate "$\uparrow K = \uparrow K'$?" for finite sets K and K' (just check $\forall k \in K\, \forall k' \in K$ the predicates $k \le k'$ and $k' \le k$). Finally, K_m is a computable finite basis of Pred$^*(I)$. $\square$

Definition 3.2. *The* covering problem *is to decide, given two states s and t, whether starting from s it is possible to cover t, i.e. to reach a state $t' \ge t$.*

The covering problem is often called the "control-state reachability problem" when S has the form $Q \times D$ (for Q a finite set of so-called "control states" and D an infinite set of data values, e.g. lossy channel systems) and $(q,d) \leq (q',d')$ entails $q = q'$ and $d \leq d'$.

Theorem 3.1. *The covering problem is decidable for WSTS's with effective pred-basis and decidable $\leq$.*

Proof. Thanks to Proposition 3.3, it is possible to compute K, a finite basis of $\text{Pred}^*(\uparrow t)$. It is possible to cover t starting from s iff $s \in\uparrow K$. By decidability of $\leq$, it is possible to check whether $s \in\uparrow K$. □

Variants of this problem can be decided in the same way. E.g. deciding whether t can be covered from all states in a given upward-closed I.

3.2 Tree-saturation methods

We speak of tree-saturation methods when we have methods representing (in some way) all possible computations inside a finite tree-like structure. In this section, we illustrate the idea with the Finite Reachability Tree and its several applications to termination, inevitability, and boundedness problems.

We assume $\mathcal{S} = \langle S, \rightarrow, \leq \rangle$ is a WSTS.

Finite reachability tree

Definition 3.3. *For any $s \in S$ the Finite Reachability Tree from s (denoted as FRT(s)), is a directed unordered tree where nodes are labelled by states of S. Nodes are either dead or live. The root node is a live node n_0, labeled by s (written $n_0 : s$). A dead node has no child node. A live node $n : t$ has one child $n' : t'$ for each successor $t' \in Succ(t)$. If along the path from the root $n_0 : s$ to some node $n' : t'$ there exists a node $n : t$ $(n \neq n')$ such that $t \leq t'$, we say that n subsumes n', and then n' is a dead node. Otherwise, n' is live.*

Thus leaf nodes in FRT(s) are exactly the nodes labelled with terminal states and the subsumed nodes.

Example 5 (FRT for Context-Free Grammars)**.** *Let's build some examples of a finite reachability tree in the context of CFG (Definition 2.7).*

Let's consider $G = \langle \{S, X, Y\}, \{a, b\}, R_G \rangle$ where R_G (the set of rules) is given by

$$\begin{aligned} S &\rightarrow YX \mid b \\ X &\rightarrow S \\ Y &\rightarrow a. \end{aligned}$$

For example, a possible derivation from starter character S to terminal word ab is given by:

$$S \to_G YX \to_G aX \to_G aS \to_G ab.$$

$$\begin{tikzcd}
 & & & & XY \arrow[dlll] \arrow[drr] & & & \\
 & SY \arrow[dl] \arrow[d] \arrow[drr] & & & & & Xa \arrow[d] & \\
bY \arrow[d] & \boxed{YXY} & & Sa \arrow[dl] \arrow[dr] & & & Sa \arrow[dl] \arrow[dr] & \\
ba & & ba & & \boxed{YXa} & \boxed{YXa} & & ba
\end{tikzcd}$$

Figure 3: FRT(XY) in $\langle S_G, \leq_P \rangle$. Subsumed nodes are boxed.

$$\begin{tikzcd}
 & & & & XY \arrow[dlll] \arrow[drr] & & & \\
 & SY \arrow[dl] \arrow[d] \arrow[drr] & & & & & Xa \arrow[d] & \\
bY \arrow[d] & \boxed{YXY} & & Sa \arrow[dl] \arrow[dr] & & & Sa \arrow[dl] \arrow[dr] & \\
ba & & ba & & YXa \arrow[dl] \arrow[dr] & \boxed{YXa} & & ba \\
 & & & aXa \arrow[d] & & \boxed{YSa} & & \\
 & & & \boxed{aSa} & & & &
\end{tikzcd}$$

Figure 4: FRT(XY) in $\langle S_G, \preceq \rangle$. Subsumed nodes are boxed

Figure 3 and Figure 4 displays FRT(XY) for $\langle S_G, \leq_P \rangle$ and FRT(XY) for $\langle S_G, \preceq \rangle$ respectively.

Lemma 3.3 (König)**.** *Let T be a rooted tree with infinite nodes, each with a finite number of children. Then T has a branch of infinite length.*

Proof. We will show that we can choose an infinite sequence of nodes $t_0, t_1, t_2, \ldots$ of T such that:

- t_0 is the root node;
- t_{n+1} is a child of t_n;
- each t_n has infinitely many descendants.

Then the sequence $t_0, t_1, t_2, \ldots$ is such a branch of infinite length. Take the root node t_0. By definition, it has a finite number of children. Suppose that all of these children had a finite number of descendants. Then that would mean that t_0 had a finite number of descendants, and that would mean T was finite. So t_0 has at least one child with infinitely many descendants. Thus, we may pick t_1 as any one of those children.

We can conclude with an induction proof: suppose node t_k has infinitely many descendants. As t_k has a finite number of children, by the same argument as above, t_k has at least one child with infinitely many descendants. Thus we may pick t_{k+1} which has infinitely many descendants. The assertion is followed by the Axiom of Dependent Choice. □

Lemma 3.4. *FRT(s) is finite.*

Proof. The wqo property ensures that all paths in $FRT(s)$ are finite because an infinite path would have to contain a subsumed node. Finite branching and König's Lemma conclude the proof.

□

With finiteness, we observe that $FRT(s)$ is effectively computable if $\mathcal{S}$ has (1) a decidable $\leq$, and (2) effective *Succ* (i.e, the Succ mapping is computable).

The construction of FRT(s) does not require compatibility between $\leq$ and $\rightarrow$. However, when we have compatibility, FRT(s) contains, in a finite form, sufficient information to answer several questions about computational paths starting from s.

Lemma 3.5. *Any computation starting from s has a finite prefix labelling maximal path in FRT(s).*

Proof. Follows immediately from the finiteness of FRT(s) (Lemma 3.4). □

Further results need slightly restricted notions of compatibility: transitive compatibility and stuttering compatibility.

To prove the decidability of the termination problem, we need this last proposition.

Proposition 3.4. *Assume $\mathcal{S}$ is a WSTS with* transitive compatibility *(Definition 2.4) $\mathcal{S}$ has a non-terminating computation starting from s iff FRT(s) contains a subsumed node.*

Proof. ($\Rightarrow$) : Consider a non-terminating computation. A finite prefix labels a path in FRT(s) (Lemma 3.5). The last node of this path is a leaf node, not labelled with a terminal state, hence a subsumed node.

($\Leftarrow$) : If $n_2 : t_2$ is the leaf node subsumed by $n_1 : t_1$, we have $s \xrightarrow{*} t_1 \xrightarrow{t} t_2$ with $t_1 \leq t_2$. Transitive compatibility allows us to infer the existence of some $t_2 \to t_3$ with $t_2 \leq t_3$. Repeating this reasoning, we build an infinite computation starting from s. □

Hence we have

Theorem 3.2. *Termination is decidable for WSTS's with transitive compatibility, decidable $\leq$ and effective Succ($\cdot$).*

Other proprieties. Assume $\mathcal{S}$ is a WSTS with *stuttering compatibility.*

Proposition 3.5. *Assume I is upward-closed. There exists a computation starting from s where all states are in I iff $FRT(s)$ has a maximal path where all nodes are labelled with states in I.*

Proof. ($\Rightarrow$) Use Lemma 3.5.

($\Leftarrow$) Assume that $n_0 : t_0, \ldots, n_k : t_k$ is a maximal path in $FRT(s)$ with all labels in I. If n_k is a live node, then $t_0 \to t_1 \to \cdots t_k$ is a computation and we are done. If n_k is a dead node, then we display an infinite computation ($s =$) $s_0 \to s_1 \to \cdots$ where all states are greater (w.r.t. $\leq$) than one of the t_i 's, and thus belong to I.

We define the s_i 's inductively, starting from $s_0 \overset{\text{def}}{=} s\,(= t_0)$. Assume we have already built $s_0, \ldots, s_n$. We have $s_n \geq t_i$ for some $i \leq k$. There are two cases:

- $i < k$: then $t_i \to t_{i+1}$. Because of stuttering compatibility, there exists a sequence $s_n \to \cdots \to s_m (m > n)$ with $s_n, \ldots, s_{m-1} \geq t_i$ and $s_m \geq t_{i+1}$. We use them to lengthen our sequence up to s_m.

- $i = k$: then, because n_k is dead, $t_j \leq t_k$ for some $j < k$. Thus $t_j \leq s_n$, so that we are back to the previous case and can lengthen our sequence.

□

The control-state *maintainability problem* is to decide, given an initial state s and a finite set $Q = \{t_1, \ldots, t_m\}$ of states, whether there exists a computation starting from s where all states cover one of the t_i 's. The dual problem (called the *inevitability problem*) is to decide whether all computations starting from s eventually visit a state not covering one of the t_i's.

Theorem 3.3. *The control-state maintainability problem and the inevitability problem are decidable for WSTS's with (1) stuttering compatibility, (2) decidable $\leq$, and (3) effective Succ.*

Proof. Thanks to Proposition 3.5, the control-state maintainability problem reduces to checking whether $FRT(s)$ has a maximal path with all labels in $\uparrow Q$. □

4 Petri nets

4.1 Introduction

Petri nets are graphical and mathematical modelling framework applicable to discrete even systems. Petri nets are promising tools for representing and studying information processing systems that are characterized as being concurrent, asynchronous, distributed, parallel, nondeterministic and stochastic. As a graphical tool, Petri nets can be used as visual communication aids similar to flow charts, block diagrams and networks. In addition, tokens are used in these nets to simulate the dynamic and concurrent activities of systems. As a mathematical formalism, it is possible to set up state equations, algebraic equations, and other mathematical models governing the behaviour of systems.

Conceived by the German mathematician and computer scientist, Carl Adam Petri, in the 1960s, Petri nets have since evolved into a robust and extensively utilized theory for addressing real-world challenges.

Petri nets have been proposed for a wide variety of applications. This is due to the generality and permissiveness inherent in Petri nets. They can be applied informally to any system that can be described graphically like flow charts and that needs some means of representing parallel or concurrent activities. However, careful attention must be paid to a trade-off between modelling generality and analysis capability. That is, the more general the model, the less amenable it is to analysis. In fact, a major weakness of Petri nets is the complexity problem, i.e. Petri-net-based models tend to become too large for analysis even for a modest-size system.

This section will provide the foundational aspects of Petri nets, it will show major generalizations, define classical theoretical problems and link to the WSTS environment. For a more gentle and complete presentation of Petri nets, you refer to [Rei13b].

4.2 Formal definition

In graphical representation, places are drawn as circles and transitions as bars or boxes. Arcs are labelled with their weights (positive integers), where a k-weighted arc can be interpreted as the set of k parallel arcs. Labels for unity weight are usually omitted. A marking (state) assigns a non-negative integer to each place. If a marking assigns to place p a non-negative integer k, we say that p is marked with k tokens. Graphically, we place k black dots (tokens) in place p. A marking is denoted by M, an m-vector, where m is the total number of places. The p-th component of M, denoted by $M(p)$, is the number of tokens in place p. In modelling, using the concept of conditions and events, places represent conditions, and transitions represent events. A transition (an event) has a certain number of input and output places representing the preconditions and postconditions of the event, respectively. The presence of a token in a place is interpreted as holding the truth of the condition associated with the place. In another interpretation, k tokens are placed to indicate that k data items or resources are available.

Definition 4.1 (Petri net)**.** *A Petri net structure is a 5-tuple $PN = \langle P, T, A, W, M_0 \rangle$ where:*

- $P = \{p_1, p_2, \dots, p_m\}$ *is a finite set of places;*
- $T = \{t_1, t_2, \dots, t_n\}$ *is a finite set of transitions;*
- $A \subseteq (P \times T) \cup (T \times P)$ *is the set of arcs (flow relation);*
- $W \colon A \to \{1, 2, 3, \dots\}$ *is a weight function;*
- $M_0 \colon P \to \{0, 1, \dots\}$ *is the initial marking;*
- $P \cap T = \emptyset$ *and* $T \cap P = \emptyset$.

A Petri net structure $N = \langle P, T, A, W \rangle$ without any specific initial marking is denoted by N.

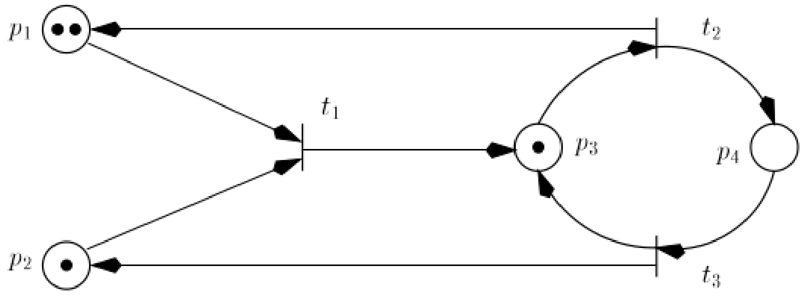

Figure 5: Graphic representation of a Petri net. t_1, t_2 are enable, t_3 is disabled.

Definition 4.2. *Let's define two preliminary concepts to understand the firing rule.*

- Enable transition*: A transition is said "enable transition" if each input place has at least one token. For all input places the number of tokens in a place of enable transition must be equal to or greater than the weight of the arc connecting the named place with the transition.*
- Disabled Transition*: We say that a transition is a "disabled transition" if there exists an input place with the number of tokens less than the weight of the corresponding arc.*

The behaviour of many systems can be described in terms of system states and their changes. In order to simulate the dynamic behaviour of a system, a state or marking in a Petri net is changed according to the following transition (firing) rule:

Definition 4.3 (Firing rule)**.** *An enabled transition may or may not fire (depending on whether or not the event actually takes place). A firing of an enabled transition t removes $w(p, t)$ tokens from each input place p of t, and adds $w(p, t)$ tokens to each output place p of t, where $w(t, p)$ is the weight of the arc from t to p.*

A transition without any input place is called a *source* transition, and one without any output place is called a *sink* transition. Note that a source transition is unconditionally enabled and that the firing of a sink transition consumes tokens, but does not produce any.

Notation. *We can denote the marking of a Petri net by using the multi-set notation. For example in Figure 4.1 we will denote the marking M as $p_1^2 p_2 p_3 p_4$ or $\{p_1, p_1, p_2, p_3\}$.*

Definition 4.4 (Common extensions). *We can define numerous extensions, here are the most common ones:*

- Petri nets with inhibitory arcs *extend the basic model with special "inhibitory" arcs (also called "zero-test" arcs) that forbid (inhibit) the firing of a given transition when a given place is not empty.*
- Petri nets with transfer arcs *extend the basic model with special "transfer" arcs. Here transitions fire as usual but their effect is richer: the transfer arcs say whether the full content of some place must be transferred (added) to some other place.*
- Petri nets with reset arcs *extend the basic model with special "reset arcs" telling how the firing of some transitions resets (empties) some places.*
- *Self-modifying nets are Petri nets where the weight on arcs is not constant anymore. Rather it is an expression evaluating into a linear combination (with non-negative coefficients) of the current contents of the places. Post self-modifying nets are self-modifying nets where the self-modifying extension is only allowed on "post" arcs (arcs from transitions to places).*

4.3 Petri nets as WSTS

The simplest ordering between markings is inclusion: $M \subseteq M'$ iff $M(p) \leq M'(p)$ for every place $p \in P$. This is a well quasi-order as proved in the Example 2. In all these extensions, reachability becomes undecidable, however:

Theorem 4.1. *Using the inclusion ordering,*

- *Petri nets are WSTS's with strong strict compatibility,*
- *Petri nets with transfer arcs are WSTS's with strong strict compatibility,*
- *Petri nets with reset arcs are WSTS's with strong compatibility,*
- *Post self-modifying nets are WSTS's with strong strict compatibility.*

Proof. Let $PN = \langle P, T, A, W, M_0 \rangle$ be any extension of a petri net. Let $\mathcal{M} := \mathbb{N}^P$ be the set of all possible markings in places P (or, equivalently, a multiset with elements of P). It's easy to show that $(\mathcal{M}, \subseteq)$ is a wqo (see Example 2). The set of transition T and arcs A can be seen as a transition relation $\rightarrow : \mathcal{M} \times \mathcal{M}$. Let's prove that $(\mathcal{M}, \subseteq, \rightarrow)$ is a WSTS with strong strict compatibility in all extensions (except reset arcs).

Let $M_1 \subset N_1$ and $M_1 \rightarrow M_2$, we want to show that there exists a transition $N_1 \rightarrow N_2$ with $M_2 \subset N_2$. With a detailed (but easy checking) it's possible to prove (in all extensions cited above, except with reset arcs) that there are no constrains about applying the firings which transformed M_1 into M_2 at the state N_1 since $M_1 \subset N_1$. We thus obtain a marking N_2 which satisfy the propriety.

In the case of reset arcs the same reasoning can be applied except for the following: since the firing can resets some places, it can happen that both M_2 obtained from $M_1 \rightarrow M_2$ and N_2 obtained from $N_1 \rightarrow N_2$ (with same firings) collapses to a same marking $M_2 = N_2$ (e.g. $\forall p \in P \; M_2(p) = N_2(p) = 0$) even if $N_1 \subset M_1$. $\square$

Proposition 4.1. *Petri Nets*[5] *have effective pred basis.*

Proof. Let $PN = \langle P, T, A, W, M_0 \rangle$ be a petri net. Let $\mathcal{M} := \mathbb{N}^P$ be the set of all possible markings in places P (or, equivalently, a multiset with elements of P).

We want to prove that exists an algorithm such that $\forall M \in \mathcal{M}$ we can return a finite basis of $\uparrow \text{Pred}(\uparrow M)$. The upward-closed set $\uparrow M$ it's obvious to calculate (it's just the set of all the markings M' with $\forall p \in P \; M(p) \leq M'(p)$). Then we have that $\text{Pred}(\uparrow M)$ is computable by just considering all possible firings (which are finite) that created a minimal marking (which are finite) of $(\uparrow M, \subseteq)$. Finally we can compute $\uparrow \text{Pred}(\uparrow M)$ as before. $\square$

So covering and termination, which are classical problems in the Petri net field, is decidable thanks to Theorem 3.1 and 3.2. This also applies to the three extensions (transfer arcs, reset arcs and post self-modifying nets) we mentioned.

Other orderings can turn Petri nets into WSTS's.

Assume $PN = \langle P, T, F, M_0 \rangle$ is a marked net (a net with a given initial marking M_0). Say a place $p \in P$ is unbounded if there are reachable (from M_0) markings with an arbitrarily large number of tokens in p. Separate bounded and unbounded places and write $P = P_b \sqcup P_{nb}$. Usually, one sees places in P_b as "control places" and places in P_{nb} as "data places" or "counter places". We can then define the ordering:

$$M \ll M' \overset{\text{def}}{\Leftrightarrow} \begin{cases} M(p) = M'(p) \text{ for all } p \in P_b, \\ M(p) \leq M'(p) \text{ for all } p \in P_{nb}. \end{cases}$$

[5]Extensions are not considered here

This is a *well-ordering over the set of reachable markings*. So that, if we associate to a marked net $\langle N, M_0 \rangle$ a transition system $\mathcal{S}_{N,M_0}$ containing only the reachable markings we get

Proposition 4.2. *$\langle \mathcal{S}_{N,M_0}, \ll \rangle$ is a WSTS.*

This works for all the extensions like post self modifying nets, etc. we mentioned earlier. However, the well-ordering is only decidable when we can tell effectively which places of the net are bounded. This can be done for Petri nets and for post self-modifying nets. (For nets with reset arcs and nets with transfer arcs, telling whether a given place p is bounded is not decidable).

The partial bounded reachability problem is, given a marked net N, M_0 and a marking M, to tell whether from M_0 it is possible to reach an M' with $M'(p) = M(p)$ for all $p \in P_b$.

Theorem 4.2. *The partial bounded reachability problem is decidable for Petri nets and post self-modifying nets.*

Proof. The partial bounded reachability problem is an instance of the covering problem for $\langle \mathcal{S}_{N,M_0}, \ll \rangle$. □

The most surprising aspect of this result is the relative simplicity of the algorithmic notions that are involved.

References

[Ott87] Thomas Ottmann. *Automata, Languages and Programming: 14th International Colloquium Karlsruhe, Federal Republic of Germany, July 13–17, 1987 Proceedings.* Jan. 1987. DOI: 10.1007/3-540-18088-5.

[Fin90] Alain Finkel. "Reduction and covering of infinite reachability trees". In: *Inf. Comput.* 89 (Dec. 1990), pp. 144–179. DOI: 10.1016/0890-5401(90)90009-7.

[AJ96] Parosh Abdulla and Bengt Jonsson. "Verifying Programs with Unreliable Channels". In: *Inf. Comput.* 127 (June 1996), pp. 91–101. DOI: 10.1006/inco.1996.0053.

[Abd+96] Parosh Abdulla et al. "General decidability theorems for infinite-state systems". In: Aug. 1996, pp. 313–321. DOI: 10.1109/LICS.1996.561359.

[Abd+04] Parosh Abdulla et al. "Using Forward Reachability Analysis for Verification of Lossy Channel Systems". In: *Formal Methods in System Design* 25 (July 2004), pp. 39–65. DOI: 10.1023/B:FORM.0000033962.51898.1a.

[Rei13a] W. Reisig. *Understanding Petri nets. Modeling techniques, analysis methods, case studies. Translated from the German by the author.* July 2013. DOI: 10.1007/978-3-642-33278-4.

[Rei13b] W. Reisig. *Understanding Petri nets. Modeling techniques, analysis methods, case studies. Translated from the German by the author.* July 2013. DOI: 10.1007/978-3-642-33278-4.

[SS14] Sylvain Schmitz and Ph Schnoebelen. "The Power of Well-Structured Systems". In: Feb. 2014. DOI: 10.1007/978-3-642-40184-8_2.