Fundamental structures in well-structured infinite transition systems

A. Finkel and Ph. Schnoebelen

Lab. Specification and Verification,* ENS de Cachan & CNRS URA 2236.

Abstract. We suggest a simple and clean definition for Well-Structured Transition Systems [Fin90,AČJY96], a general class of infinite state systems for which decidability results exist. As a consequence we can (1) generalize the definition in many ways, (2) find examples of (general) WSTS's in many fields, and (3) present new decidability results.

Introduction

Formal verification of programs and systems is a very active field for both theoretical research and practical developments, especially since impressive advances in formal verification technology proved feasible in several realistic applications from the industrial world. The highly successful model-checking approach for finite systems [BCM+92] suggested that a working verification technology could well be developed for systems with an infinite state space.

This explains the considerable amount of work that has been spent in recent years on this Verification of Infinite State Systems (VISS) field, with a surprising wealth of positive results [Mol96,Esp97]. The field now has its own conference.

However, this wealth of positive results is quite disorganized. Many people investigated extensions of essentially finite formal models for which decidability results existed, and they had to carefully control which kind of extensions could be afforded. A very interesting development in this field is the introduction of *well-structured transition systems* (WSTS's). These are transition systems where the existence of a well-quasi-ordering over the infinite set of states ensures the termination of several algorithmic methods. WSTS's are an abstract generalization of several specific structures and they allow general decidability results that can be applied to Petri nets, lossy channel systems, and many more. (Of course, WSTS's are not intended as a general explanation of all the decidability results one can find for specific models.)

Finkel [Fin87a,Fin87b,Fin90] was the first to propose a definition of WSTS (actually several variant definitions). His insights came from the study of Petri nets where several decidability results rely on a monotonicity property (transitions firable from marking M are firable from any larger marking) and Dickson's lemma (inclusion between markings of a net is a well-ordering). He mainly

* 61, av. Pdt Wilson; 94235 Cachan Cedex; FRANCE. email: {finkel,phs}@lsv.ens-cachan.fr.

investigated the decidability of termination, boundedness and coverability-set problems. He applied the idea to several classes of fifo nets and of CFSM's (see Section 10).

Independently, Abdulla *et al.* [AČJY96] later proposed another definition. Their insights came from their study of lossy-channel systems and other families of analyzable infinite-state systems (e.g. integral relational automata [Čer94]). They mainly investigated covering, inevitability and simulation problems. They applied the idea to timed networks [AJ97] and lossy systems.

Later, Kushnarenko and Schnoebelen [KS97] introduced WSTS's with *downward* compatibility, motivated by some analysis problems raised by Recursive-Parallel Programs.

In this paper, we propose a simpler and cleaner definition of WSTS's by separating structural and effectiveness issues. (For clarity, we do not consider *labeled* transitions and only focus on core problems related to reachability and inevitability of sets of states.) We also show how this basic definition can be *generalized* in many ways, allowing many systems to be seen as (some kind of) WSTS. Indeed we show, through a large collection of examples, that WSTS's are ubiquitous provided the key notion, *compatibility of transitions with a well-ordering*, is explored under various angles.

The following table collects all examples we mention in the paper.

	upward	downward	strict	non-strict	strong	reflexive	stuttering	transitive	
Petri nets	•		•	○	•	○	○	○	Petri Nets and their extensions
post SM-nets	•		•	○	•	○	○	○	
Petri nets with transfer arcs	•		•	○	•	○	○	○	
Petri nets with reset arcs	•			•	•	○	○	○	
BPP with inhibitory arcs		•		•		•			
CFSM's with Lossy Channels	•			•			•	○	Communicating Finite State Machines
CFSM's with Insertion Errors		•		•			•	○	
Monogeneous CFSM's [Fin82]	•		•	○	•	○	○	○	
Free-choice FIFO nets	•			•			•	○	
Synchronizable CFSM's	•		•	○	•	○	○	○	
BPA (Basic Process Algebra)		•		•		•			Process Algebras
BPP (Basic Parallel Process)	•		•	○	•	○	○	○	
Normed BPA	•			•			•	○	
Context-free grammars	•		•	○	•	○	○	○	String Rewriting
Permutative grammars		•		•		•			
RPPS Schemes		•		•		•			Other models
RPPS Schemes with $\preceq_\perp$	•			•			•	○	
Integral Relational Automata [Čer94]	•			•	•	○	○	○	

(A • indicates presence, a ○ is presence implied by a stronger •.)

The paper uses *Petri nets* (§ 2) to introduce *WSTS's* (§ 3) because we first exemplify the *Finite Reachability Tree* algorithm (§ 4). Boundedness analysis leads to *strict WSTS's* (§ 5). Then § 6 presents the *Saturation method* for reachability analysis. Other WSTS's are found in *string rewriting* (§ 7) and *simple process algebra* (§ 7). This motivates the introduction of *stuttering WSTS's* (§ 9). Other WSTS's are found in *Communicating Finite State Machines* (§ 10), motivating the introduction of *downward WSTS's* (§ 11).

1 Transition Systems

A *transition system* (TS) is a structure $\mathcal{S} = \langle S, \rightarrow, \ldots\rangle$ where $S = \{s, t, \ldots\}$ is a set of *states*, and $\rightarrow \subseteq S \times S$ is any set of *transitions*. TS's may have additional structure like initial states, labels for transitions, durations, causal independence relations, etc., but in this paper we are only interested into the state-part of the behaviors.

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 *predecessors*). A state with no successor is a *terminal state*. $\mathcal{S}$ is *finitely branching* if all $Succ(s)$ are finite.

2 The well-structure of Petri nets

Petri nets are a well-known model of concurrent systems giving rise to infinite TS's. Still, many questions about the behavior of Petri nets (e.g. reachability, finiteness, ...) can be decided.

A key ingredient in methods for the analysis of Petri nets is the *monotonicity* of the firing rule. We describe this upon the example net from Figure 1. The current marking, denoted by the four tokens inside the places, is

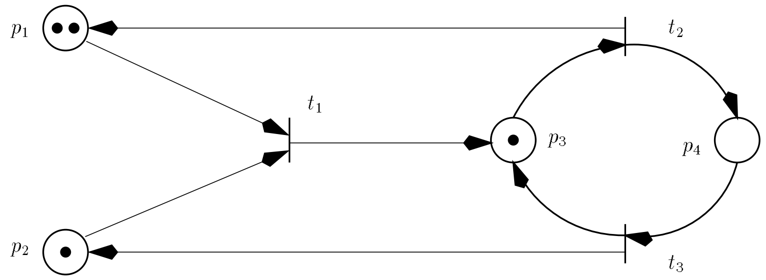

Fig. 1. A Petri net

$M_0 \stackrel{\text{def}}{=} \{p_1, p_1, p_2, p_3\}$ (also denoted by $p_1^2 p_2 p_3$). From M_0, t_1 and t_2 are firable. Firing t_2 from M_0 leads to $M_1 \stackrel{\text{def}}{=} \{p_1, p_1, p_1, p_2, p_4\}$.

Now, if we had started from a marking M_0' that includes M_0, then t_1 and t_2 are still firable (and maybe t_3 also is now firable). Firing t_2 from M_0' will lead to an M_1' with the property that the difference $M_0' - M_0$ is preserved and equals $M_1' - M_1$.

This is the *monotonicity property of Petri nets*: transitions are compatible with inclusion between markings:

$$\big(M_1 \rightarrow M_2 \text{ and } M_1 \subseteq M_1'\big) \text{ entail } \big(M_1' \rightarrow M_2' \text{ and } M_2' - M_2 = M_1' - M_1\big)$$

Another key ingredient in methods for the analysis of Petri nets is that inclusion between markings is a *well-ordering*:

Definition 1. A ordering relation $\leq$ (over some set X) is a well-ordering iff, for any infinite sequence $x_0, x_1, x_2, \ldots$ in X, there exists indexes $i < j$ s.t. $x_i \leq x_j$.

If $\leq$ is a well-ordering, then any infinite sequence contains an infinite increasing subsequence: $x_{i_0} \leq x_{i_1} \leq x_{i_2} \ldots$.

3 Well-structured transition systems

The monotonicity property for Petri nets motivated the original definition of well-structured systems [Fin90]. Here we give a simplified version:

Definition 2. A well-structured transition system (WSTS) is a TS $\mathcal{S} = \langle S, \rightarrow, \preceq\rangle$ equipped with an ordering $\preceq \subseteq S \times S$ between states s.t.

- $\preceq$ is a well-ordering, and
- $\preceq$ is "compatible" with $\rightarrow$,

where "compatible" means that for all $s_1 \preceq t_1$ and transition $s_1 \rightarrow s_2$, there exists a transition $t_1 \rightarrow t_2$ s.t. $s_2 \preceq t_2$.

Thus compatibility states that $\preceq$ is a simulation relation in the Hennessy-Milner sense [HM85]. See Figure 2 for a diagrammatic presentation of compatibility where we quantify universally over solid lines and existentially over dashed lines. With $\subseteq$, Petri nets give rise to well-structured transition systems.

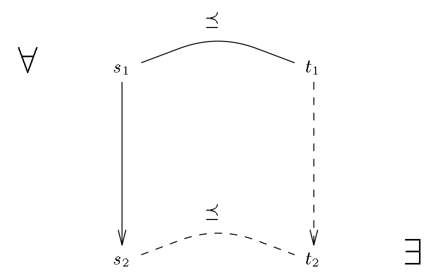

Fig. 2. Strong compatibility

4 Finite reachability tree

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

Definition 3. [Fin90] $FRT_{\preceq}(s)$, the *Finite Reachability Tree from s*, is a directed unordered tree where nodes are labeled 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 children $n' : t'$ 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 $n : t$ ($n \neq n'$) s.t. $t \preceq t'$, 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 (1) the nodes labeled with terminal states, and (2) the subsumed nodes. Fig. 3 shows $FRT(M_0)$ for our previous Petri net example. (We decorated $FRT(M_0)$ with some explanations like tran-

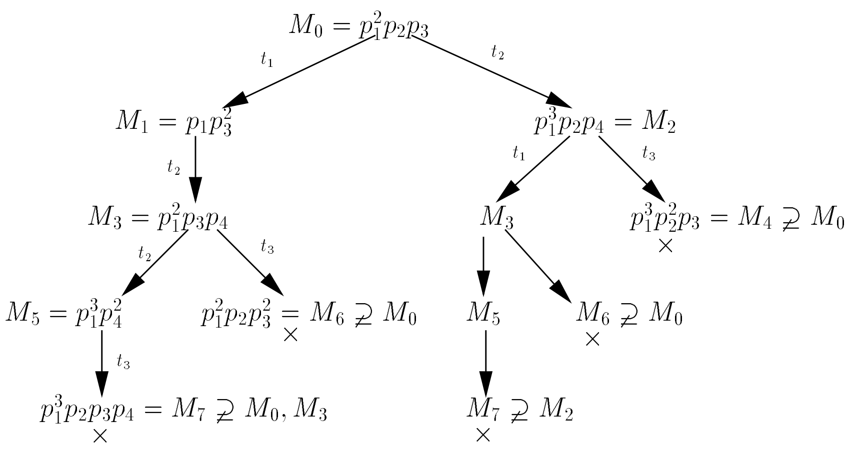

Fig. 3. Finite Reachability Tree for the Petri net example

sition names, subsumption data, and crosses for dead nodes.)

For a WSTS $\mathcal{S}$, the well-ordering property ensures that all paths in $FRT(s)$ are finite because any infinite path would have to contain a subsumed node. Now König's lemma entails that if $\mathcal{S}$ is finitely branching, then $FRT(s)$ is finite (hence the name). So that $FRT(s)$ is effectively computable if (1) $\preceq$ is decidable, and (2) the $Succ$ mapping is computable ("$\mathcal{S}$ has effective $Succ$"). Clearly Petri nets have a decidable $\subseteq$ and effective $Succ$.

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

Assume $I \subseteq S$ is *upward-closed*, i.e. $s \succeq t \in I$ entails $s \in I$. For Petri nets, I can be e.g. "all markings where a given place is marked", or "all markings where a given transition is enabled", etc.

Proposition 4. *$\mathcal{S}$ admits a computation starting from s where all states are in a given upward-closed I iff $FRT(s)$ has a maximal path where all states are in I.*[1]

When $\preceq$ is a well-ordering, any upward-closed I can be represented via a finite *basis*, i.e. as "$I = \{s \in S \mid s \succeq s_1 \vee \ldots \vee s \succeq s_k\}$". When $\mathcal{S}$ has effective *Succ* and decidable $\preceq$ and $I \subseteq S$ is effectively given via a finite basis, the FRT can be used to yield:

Proposition 5. [AČJY96] *CSM, the Control State Maintainability problem, is decidable for WSTS's with effective Succ and decidable $\preceq$.*

where CSM inputs some state s and a finite basis I_f, and answers yes iff there exists a computation starting from s where all visited states are in I (i.e. I can be maintained), a property written $s \models \exists\Box I$ in temporal logic [2].

A dual view is possible: assume $D \subseteq S$ is downward-closed.

Proposition 6. [AČJY96] *DInev, the Inevitability problem, is decidable for WSTS's with effective Succ and decidable $\preceq$.*

where DInev inputs some state s and a finite co-basis for D (i.e. a basis for $S \setminus D$), and answers yes iff all computations starting from s eventually reach a state in D (i.e. D is inevitable), a property written $s \models \forall\Diamond I$ in temporal logic. Termination is a special case of DInev because the set of terminal states is downward-closed.

Looking back to our Petri net from Fig. 1, $FRT(M_0)$ lets us see that $M_0 \models \exists \uparrow p_2$, i.e. it is possible to have p_2 always marked. One just has to check all paths and notice the rightmost one: M_0, M_2, M_4.

5 Strict compatibility

In Petri nets the monotonicity property is much stronger than just "compatibility". Finkel [Fin90] also considered WSTS's with *strict compatibility*. Strict compatibility is a stronger form of compatibility. It is given by using the strict $\prec$ rather than $\preceq$ in Definition 2.

Hence strict compatibility means that from strictly larger states it is possible to reach strictly larger states. See Figure 4 for a diagrammatic presentation. Many extensions of Petri nets are WSTS's with strict compatibility: e.g. finite

[1] where we shortly say that $FRT(s)$ contains "a state" when we mean "a node labeled by a state".

[2] If now we have another upward-closed I' (effectively given) and if I does not contain terminal states, then $I' \models \exists\Box I$ is decidable. However when I contains terminal states, $I' \models \exists\Box I$ is not decidable in general.

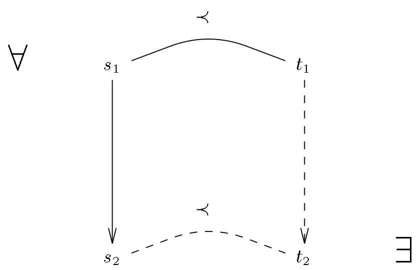

Fig. 4. Strict compatibility

colored Petri nets, or Valk's post-SM nets [Val78]. Other extensions of Petri nets [Cia94,LC94] allow special kind of arcs, e.g. *reset arcs* which empty the attached place when the corresponding transition is fired, *transfer arcs* which move all tokens from a given place to another place, and *copy arcs* which add a copy of every tokens from some place to a given place. All these extensions enjoy compatibility in the strict sense, except for nets with reset arcs which only have non-strict compatibility. Other extensions like zero-test arcs or inhibitory arcs clearly break the compatibility requirement (w.r.t. $\subseteq$).

Effective WSTS's with strict compatibility have a decidable boundedness problem:

Proposition 7. [Fin90] *If $\mathcal{S}$ is a finitely branching WSTS with strict compatibility, $Succ^*(s)$ is infinite iff $FRT(s)$ contains a leaf node $n : t$ subsumed by an ancestor $n' : t'$ with $t' \prec t$.*

Corollary 8. *The boundedness problem is decidable for strict-WSTS's with (1) decidable $\preceq$ and (2) computable Succ.*

As an example, $FRT(M_0)$ has a leaf strictly subsumed by one of its ancestors: we conclude that, starting from M_0, one can reach an infinite number of different markings.

For the boundedness problem, it is possible to follow ideas from [Fin93] and build a smaller tree

Definition 9. $RRT(s)$, the *Reduced Reachability Tree*, is a tree built like $FRT(s)$ except that now a node $n : t$ is dead as soon as it has an ancestor node $n' : t'$ with $t' \preceq t$ or $t \preceq t'$.

Theorem 10. *If $\mathcal{S}$ is a finitely branching WSTS with strict compatibility, $Succ^*(s)$ is infinite iff $RRT(s)$ contains a leaf node $n : t$ subsumed by an ancestor $n' : t'$ with $t' \prec t$.*

In our Fig. 3 example, $RRT(M_0)$ equals $FRT(M_0)$.

6 Saturation methods

We speak of *saturation methods* when we have methods whose termination relies on the following Lemma:

Lemma 11. *Assume $\preceq$ is a well-ordering. Then any infinite increasing sequence $I_0 \subseteq I_1 \subseteq I_2 \subseteq \cdots$ of upward-closed sets eventually stabilizes, i.e. there is a $k \in \mathbb{N}$ s.t. $I_k = I_{k+1} = I_{k+2} = \ldots$*

For a WSTS $\mathcal{S}$, a saturation method can be used to compute the set $Pred^*(I)$ of all states from which one can reach a state in I. This relies on the following consequence of the compatibility property:

Proposition 12. *If $I \subseteq S$ is upward-closed, then $Pred^*(I)$ is upward-closed.*

and in fact all $\bigcup_{i=0,\ldots,k} Pred^i(I)$ are upward-closed, so that, thanks to Lemma 11, the sequence $J_0 \subseteq J_1 \subseteq \cdots$ given by $J_0 \stackrel{\text{def}}{=} I$ and $J_{n+1} \stackrel{\text{def}}{=} J_n \cup Pred(J_n)$ eventually stabilizes at some J_k. Then $J_k = Pred^*(I)$. (Observe that stabilization is ensured as soon as two consecutive sets are equal.)

As an example, consider the Petri net from Fig. 1 and write I for $\uparrow M_6$, i.e. all markings covering $M_6 = p_1^2 p_2 p_3^2$ (for $R \subseteq S$, $\uparrow R$ denotes $\{s \in S \mid \exists s' \in R, s \succeq s'\}$, the *upward-closure* of R). Using

$$Pred(R \cup R') = Pred(R) \cup Pred(R') \tag{1}$$
$$Pred(\uparrow R) = \uparrow Pred(R) \tag{2}$$

we compute $Pred^*(I)$ by the saturation method:

$$\begin{aligned}
J_0 &= \uparrow M_6 \\
J_1 &= J_0 \cup Pred(\uparrow M_6) = \uparrow M_6 \cup \uparrow M_1 \cup \uparrow M_2 \\
J_2 &= J_1 \cup Pred(\uparrow M_6 \cup \uparrow M_1 \cup \uparrow M_2) \\
&= J_1 \cup \uparrow Pred(M_6) \cup \uparrow Pred(M_1) \cup \uparrow Pred(M_2) \\
&= J_1 \cup \uparrow \{p_1^2 p_4, p_1^3 p_2^2\} \cup \uparrow M_0 \cup \uparrow M_0 \\
&= \uparrow M_0 \cup \uparrow M_1 \cup \uparrow M_2 \cup \uparrow p_1^2 p_4 \cup \uparrow p_1^3 p_2^2 \\
J_3 &= J_2 \cup Pred(J_2) \\
&= J_2 \cup \uparrow Pred(p_1^2 p_4) \cup \uparrow Pred(p_1^3 p_2^2) \\
&= J_2 \cup \uparrow p_1 p_3 \\
J_4 &= J_3 \cup Pred(J_2) \cup Pred(\uparrow p_1 p_3) \\
&= J_3 \cup \uparrow p_1^2 p_2 \\
J_5 &= J_4 \cup Pred(J_3 \cup \uparrow p_1^2 p_2) \\
&= J_3 \cup \uparrow p_1^2 p_2 \cup \uparrow \underbrace{Pred(p_1^2 p_2)}_{=\emptyset} = J_4
\end{aligned}$$

For Petri nets, computing $Pred^*(I)$ is quite simple as we just saw. We obtain a finite basis (or a *set of residuals* according to [VJ85]'s terminology). It can be used to answer coverability questions because it is possible to cover M starting from M_0 iff $M_0 \in Pred^*(\uparrow M)$.

This algorithmic idea can be generalized for an arbitrary WSTS $\mathcal{S}$:

Definition 13. $\langle \mathcal{S}, \preceq \rangle$ has *effective pred_basis* when there is an algorithm computing a finite basis of $\uparrow Pred(\uparrow s)$ for any s.

E.g. Petri nets have effective *pred_basis*. The extensions we mentioned (reset arcs, post-SM nets, ... all have effective *pred_basis*). We made Definition 13 more general than the corresponding notion in [AČJY96] so that it also works with stuttering and transitive compatibility (& 9).

Now assume an upward-closed I is given through a finite basis I_f. We compute a sequence $J_{0,f}, J_{1,f}, \ldots$ of finite sets with $J_{0,f} \stackrel{\text{def}}{=} I_f$ and $J_{k+1,f} \stackrel{\text{def}}{=} J_{k,f} \cup pred\_basis(J_{k,f})$. We stop when $\uparrow J_{k+1,f} = \uparrow J_{k,f}$ which can be decided by checking that for any $s \in J_{k+1,f}$ we can find a $t \in J_{k,f}$ with $t \preceq s$. Then $\uparrow J_{k,f} = Pred^*(I)$ (and in fact any $J_{l,f}$ is a basis for $\cup_{i=0,\ldots,l} Pred^i(I)$). Hence the following

Proposition 14. *A finite basis for $Pred^*(I)$ can be computed for WSTS's with (1) decidable $\preceq$ and (2) effective pred_basis.*

Corollary 15. *For WSTS's with (1) decidable $\preceq$ and (2) effective pred_basis, the covering problem is decidable.*[3]

where the covering problem, called *control-state reachability* in [AČJY96], is to decide whether $s \models \exists\Diamond \uparrow I$ for a state s and an upward-closed I (given via a finite basis I_f). The variant problem "does $I' \models \exists\Diamond \uparrow I$?" also is decidable. The variant "does $D \models \exists\Diamond \uparrow I$?" is decidable (where D is downward-closed) if $\preceq$ is *intersection-effective*, i.e. there is an algorithm computing a finite basis for $(\uparrow s) \cap (\uparrow s')$ given any $s, s' \in S$.

7 The well-structure of string rewrite systems

Definition 2 is meant to abstract the fundamental requirements behind the existence of e.g. the reduced reachability tree (see Section 4). The interest of this abstract definition is that many classes of infinite TS's share this well-structure.

Consider Context-Free Grammars (CFG's) like the following example

$$G: \quad \begin{array}{l} S \to XY \mid aSS \\ X \to \epsilon \mid bXS \\ Y \to aX \mid b \end{array}$$

A possible derivation in G is

$$S \to aSS \to aXYS \to aYS \to aYXY \to aYXb \to aYb \to abb \tag{3}$$

[3] Surprisingly, to our knowledge, all covering algorithms for Petri nets in the literature are much more complicated than the simple iterated *pred_basis* approach.

If instead of focusing on the language generated by G we emphasize the rewrite steps, then G gives rise to a transition system $\mathcal{S}_G$ where states are words in $(T_G \cup N_G)^*$ and (3) now is a complete execution of $\mathcal{S}_G$, starting from S.

Let us now consider word-embedding: we say a word u embeds into a word v, written $u \preceq v$, iff u can be obtained by erasing letters from v. This gives a partial ordering, known (Higman's Lemma) to be a well-ordering.

Now assume $u \preceq v$ with $u, v \in (T_G \cup N_G)^*$. If $u \to_G u'$ by some rule, then clearly the same rule can be applied in v in such a way that $v \to_G v'$ with $u' \preceq v'$. Furthermore, if $u \prec v$ then $u' \prec v'$. So that $\langle \mathcal{S}_G, \preceq \rangle$ is a well-structured system with strict-compatibility.

Fig. 5 displays $FRT(S)$ for $\langle \mathcal{S}_G, \preceq \rangle$.

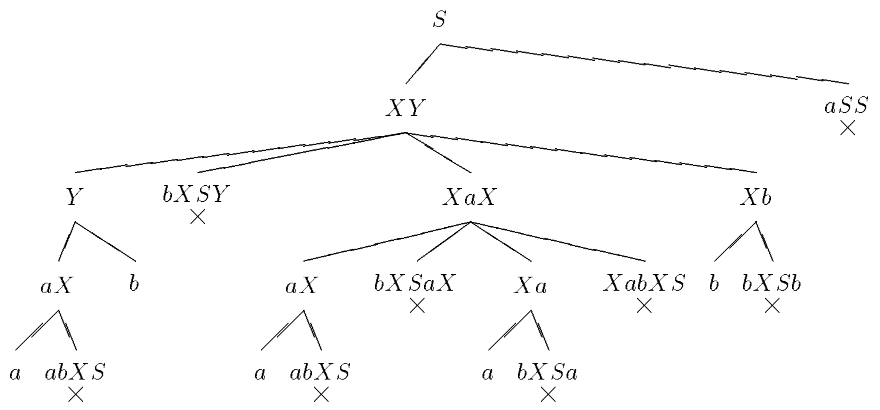

Fig. 5. $FRT(S)$ in $\langle \mathcal{S}_G, \preceq \rangle$

Another well-structured view is possible: if we consider words as multisets of symbols (rather than strings), we can define word inclusion: write $u \subseteq v$ when u can be obtained from v by a combination of erases and permutations. This yields another well-ordering (in fact, a quasi-ordering) and $\langle \mathcal{S}_G, \subseteq \rangle$ is another WSTS with strict-compatibility. When words are seen as sets of symbols rather than multisets, $\langle \mathcal{S}_G, \subseteq \rangle$ is still well-structured, but not with strict-compatibility anymore.

When it comes to algorithms for the analysis of WSTS's, the precise choice of which ordering we consider is quite relevant because many decidable properties for WSTS's (e.g. coverability) are expressed in terms of the ordering itself. When a choice is possible, using a larger ordering will often yield less information but more efficient algorithms.

8 The well-structure of Basic Process Algebra

Basic Process Algebra (BPA) is a subset of process algebra first studied in [BBK87] where only prefixing, non-deterministic choice, sequential composition and guarded recursion are allowed Here is an example BPA declaration:

$$\Delta : \quad \begin{array}{l} X \to aYX + bX + c \\ Y \to bXX + a \end{array}$$

and a possible derivation is

$$X \xrightarrow{a}_{\Delta} YX \xrightarrow{b}_{\Delta} XXX \xrightarrow{c}_{\Delta} XX \xrightarrow{c}_{\Delta} \cdots \tag{4}$$

BPA systems can be seen as CFG's with head-rewriting. (We lose head-rewriting when we replace sequential composition by parallel composition, yielding the Algebra of Basic Parallel Processes (BPP) from [Chr93], which are a subclass of Petri nets.)

Because of the head-rewriting strategy, BPA systems do not have transitions compatible with word-embedding. E.g. if in the previous example, we consider $Y \preceq XXYXX$ and step $Y \to bXX$, we cannot find some v' with $XXYXX \to v'$ and $bXX \preceq v'$. (The left-factor ordering is compatible with head-rewriting but it is not a well-ordering.)

However, if we restrict ourselves to Normed BPA (a class first introduced in [BBK87]) we can find a well-structure. Formally, a BPA process is *normed* if it admits a terminating behavior. A BPA declaration is normed if all its processes are normed. From a CFG viewpoint, this corresponds to grammars in Greibach normal form and *without useless productions*. E.g. our example Δ above is a normed BPA declaration.

With Normed BPA, the difficulty with head-rewriting can be circumvented. Let's consider $Y \preceq XXYXX$. because Δ is normed, there exists a terminating sequence $X \to \cdots \to \epsilon$. So that $XXYXX \to \cdots XYXX \to \cdots YXX \to bXXXX$ reaching $bXXXX$ with $bXX \preceq bXXXX$.

Definition 16. [KS97] A well-ordering $\preceq$ over states of some TS $\mathcal{S}$ has *stuttering compatibility* if for all $s_1 \preceq t_1$ and transition $s_1 \to s_2$, there exists a non-empty sequence $t_1 \to t_2 \to \cdots \to t_n$ with $s_2 \preceq t_n$ and $s_1 \preceq t_i$ for all $i < n$.

See Figure 6 for a diagrammatic presentation of stuttering compatibility. Observe that requiring $s_1 \preceq t_i$ makes stuttering stronger than compatibility for $\to^+$, the transitive closure of $\to$.

Proposition 17. *For a Normed BPA declaration Δ, $\langle \mathcal{S}_\Delta, \preceq \rangle$ is a WSTS with stuttering compatibility.*

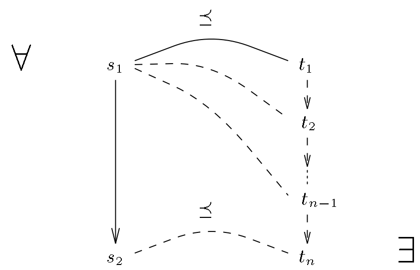

Fig. 6. Stuttering compatibility

9 Stuttering compatibility

The name "stuttering" compatibility comes from "stuttering" (also "branching") bisimulation [BCG88]. It is weaker than the usual "strong" compatibility but stronger than *transitive compatibility*, a notion we obtain when we do not require $s_1 \preceq t_i$ for $1 < i < n$. Finkel's notion of 3-structured [Fin90] is similar to transitive compatibility (in a framework where labels of transitions are taken into account). This class of WSTS's contain e.g. our RPPS schemes [KS97] and Free-Choice Fifo nets [FC87]. Of course there exists a notion of strict-stuttering (and strict-transitive) compatibility when $\prec$ instead of $\preceq$ is considered.

The decision methods we presented earlier accommodate stuttering compatibility:

Theorem 18. *1. Proposition 4 and decidability of CSM and DInev generalize to WSTS's with stuttering compatibility.*
2. Proposition 12, 14 and decidability of Coverability generalize to WSTS's with transitive-reflexive stuttering.
3. Proposition 7 and decidability of Boundedness generalize to WSTS's with strict-transitive stuttering.

These decidability results do not require any additional effectiveness hypothesis. Also, the same methods (building $FRT(.)$, iterating *pred_basis*, ...) are used with no modification.

10 The well-structure of Communicating Finite State Machines

A *Communicating Finite State Machine* (CFSM) [Boc78] can be seen as a Finite State Automaton (FSA) equipped with a collection $c_1, \ldots, c_n$ of n fifo channels. A transition of the FSA is labeled with a *send actions* (e.g. "$q \overset{c_i!a}{\rightarrow} q'$") or a *receive action* (e.g. "$q \overset{c_i?a}{\rightarrow} q'$").

A state (or configuration) of a CFSM is some $\sigma = \langle q, w_1, \ldots, w_n \rangle$ where q is a control state of the FSA, and each w_i is a word describing the current content of channel c_i. In configuration $\sigma = \langle q, w_1, \dots, w_n\rangle$ transition $q \stackrel{c_i!a}{\rightarrow} q'$ is possible, reaching $\sigma' = \langle q', w_1, \dots, w_{i-1}, w_i.a, w_{i+1}, \dots, w_n\rangle$, a new configuration where the control state is now q' and where the sent symbol a has been appended after w_i. In σ, transition $q \stackrel{c_i?a}{\rightarrow} q'$ is only possible if channel c_i contains an a in first position, i.e. if w_i is some $a.w'$. Then we can reach $\langle q', w_1, \dots, w_{i-1}, w', w_{i+1}, \dots, w_n\rangle$.

Fig. 7 shows an example where P_1 and P_2 are two different automata communicating via two fifo channels. This gives a CFSM C_{P_1,P_2} if we see P_1 and P_2 as one single global FSA.

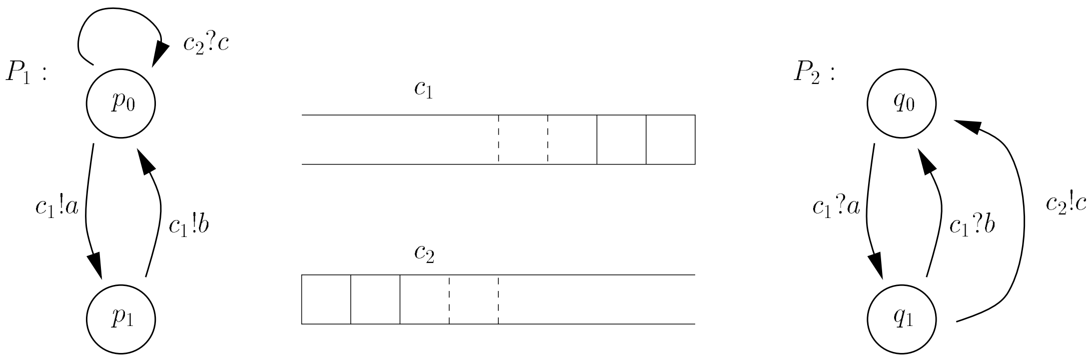

Fig. 7. A Communicating Finite State Machine

A possible behavior for this example is

$$\langle p_0q_0, \epsilon, \epsilon\rangle \stackrel{c_1!a}{\rightarrow} \langle p_1q_0, a, \epsilon\rangle \stackrel{c_1!b}{\rightarrow} \langle p_0q_0, a.b, \epsilon\rangle \stackrel{c_1?a}{\rightarrow} \langle p_0q_1, b, \epsilon\rangle \stackrel{c_2!c}{\rightarrow} \langle p_0q_0, b, c\rangle \rightarrow \cdots$$

Because the channels are unbounded, CFSM's generate infinite TS's $\mathcal{S}_C$ and are Turing-powerful. Recently, Abdulla and Jonsson investigated *Lossy Channel Systems* [AJ93,AJ94], i.e. CFSM's where in any configuration the system may loose any symbol from any channel. I.e. any transition $\langle q, w_1, \dots, w_n\rangle \rightarrow \langle q, w_1, \dots, w_i', \dots, w_n\rangle$ is possible when w_i' is obtained by removing one symbol from w_i. CFSM's with lossy channels [4] are useful as models of systems assuming unsafe communication links, e.g. the alternating bit protocol.

Several interesting decidability results for Lossy CFSM's can be explained by their well-structure [AČJY96]. Define an ordering between configurations by

$$\langle q, w_1, \dots, w_n\rangle \preceq \langle q', w_1', \dots, w_n'\rangle \text{ iff } \begin{cases} q = q' \text{ and} \\ w_i \preceq w_i' \text{ (for } i = 1, \dots, n) \end{cases}$$

The right point of view for Lossy CFSM's is to use stuttering compatibility:

[4] Finkel's *Completely Specified Protocols* [Fin94] are Lossy CFSM's where only the head symbol can be lost. They behave essentially like Lossy CFSM's.

Proposition 19. *For C a Lossy Channel System (or a CSP), $\langle \mathcal{S}_C, \preceq \rangle$ is a WSTS with (non-strict) stuttering compatibility.*

Several other special classes of CFSM's from the literature have an underlying well-structure, often with more involved orderings:

- C_{P_1,P_2} from Fig. 7 has transitions compatible with $\sqsubseteq$, defined by

$$\langle p_i q_j, w_1, w_2 \rangle \sqsubseteq \langle p_{i'} q_{j'}, w_1', w_2' \rangle \stackrel{\text{def}}{\Leftrightarrow} \begin{cases} p_i = p_{i'} \& q_j = q_{j'} \\ w_1' \in w_1.(ab)^* \text{ if } i = i' = 0, \\ w_1' \in w_1.(ba)^* \text{ if } i = i' = 1. \\ w_2' \in w_2.c^* \end{cases}$$

  This variation around the prefix ordering is not a well-ordering in general, but it is a well-ordering on $Q_1 \times Q_2 \times (ab)^* + (ab)^* a \times c^*$, a set containing all the reachable states of C_{P_1,P_2}. The same approach can be generalized to all Monogeneous CFSM's [Fin82].
- Cécé used a complicated ordering [Céc93] to show that Synchronizable CFSM [GR85] are strict WSTS's.

Cécé et al. introduced CFSM's with *insertion errors* [CFP95]. These are CFSM's where at any time, arbitrary symbols (noise) can be inserted anywhere in the channels. These too can be seen as well-structured systems, but not as easily as Lossy CFSM's.

One way is to consider the transition relation backward: When we consider the transition relation backward $\rightarrow^{-1}$, CFSM's with Insertion Errors are exactly Lossy CFSM's. This view can be useful for reachability analysis, and it helps understand why [CFP95] considered forward analysis on CFSM's with Insertion Errors, rather than the usual backward analysis based on iterated *pred_basis*.

Another approach is to consider *downward-compatibility* first introduced in [KS97] for the RPPS model.

Definition 20. [KS97] A well-ordering $\preceq$ over states of some TS $\mathcal{S}$ has *downward compatibility* if for all $s_1 \succeq t_1$ and transition $s_1 \rightarrow s_2$, there exists a $t_1 \rightarrow t_2$ with $s_2 \succeq t_2$.

See Figure 8 for a diagrammatic presentation of downward compatibility.

Proposition 21. *For C, a CFSM with Insertion Errors, $\langle \mathcal{S}_C, \preceq \rangle$ is a WSTS with (non-strict) stuttering downward compatibility.*

11 Downward compatibility

Downward compatibility is not one more generalization or specialization of the usual (upward) WSTS notion. It is a different kind of well-structure. However it had similar motivations and similar ubiquity.

Ubiquity is clear:

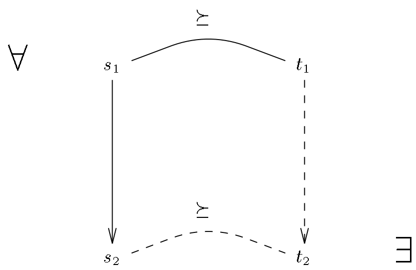

Fig. 8. Downward compatibility

1. With the usual ordering BPP nets with inhibitory arcs (even copy arcs) have reflexive downward compatibility.
2. With the usual ordering, BPA and RPPS [KS97] schemes have reflexive downward compatibility without any normedness hypothesis.
3. With the usual ordering, *permutative grammars*[5] [Mäk85] have reflexive downward compatibility.

Downward WSTS have not yet been explored in great depth. Still, some decision results exist:

Proposition 22. *Assume $\mathcal{S}$ is a downward-WSTS (possibly reflexive and/or transitive), and $D \subseteq S$ is downward-closed, then $Pred^*(D)$ is downward-closed.*

Proof. Assume $s_0 \in Pred^*(D)$. Then there is a $s_0 \to s_1 \to \cdots s_n$ with $s_n \in D$. If now $s'_0 \preceq s_0$ then downward-compatibility entails the existence of some $s'_0 \to s'_1 \to \cdots s'_m$ with $s'_m \preceq s_n$. Hence $s'_m \in D$. So that $s'_0 \in Pred^*(D)$.

Now assume an upward-closed I is given through a finite basis I_f. We define a finite sequence $J_0, J_1, \dots$ of finite sets by $J_0 \stackrel{\text{def}}{=} I_f$ and $J_{i+1} \stackrel{\text{def}}{=} J_i \cup Succ(J_i)$. We end the sequence at the first k such that $\uparrow J_k = \uparrow J_{k+1}$, a condition which is bound to hold eventually.

Proposition 23. *If $\mathcal{S}$ has downward compatibility (possibly reflexive), then $\uparrow J_k = \uparrow Succ^*(I)$.*

Corollary 24. *A finite basis for $Succ^*(I)$ can be computed for downward (possibly reflexive) WSTS's with (1) decidable $\preceq$ and (2) effective Succ.*
The sub-covering problem is decidable for these WSTS's.

where the sub-covering problem asks whether from some s one can reach a state covered by some s'.

[5] i.e. grammars where context-sensitive permutative rules $AB \to BA$ are allowed.

Conclusion

In this paper, we clarified the WSTS concept: they are TS's where transitions are "compatible" with a well-ordering. We presented a large collection of WSTS's, which suggested several variations around the idea of "compatibility", for which general decidability results were presented, some of them quite new, the other being slight extensions of earlier results.

Our own conclusion is that WSTS's are ubiquitous and can be found in many models of computation, provided "compatibility" is understood in a liberal way. Some points worth stressing are

- The WSTS idea is a guide for relating and unifying many approaches in many different fields of computer science.
- It suggests directions for extending already known decidability results.
- Dually, our own work (currently in progress) proves that investigating the limits of WSTS is a powerful guide for undecidability results.

References

[AČJY96] P. A. Abdulla, K. Čerāns, B. Jonsson, and T. Yih-Kuen. General decidability theorems for infinite-state systems. In *Proc. 11th IEEE Symp. Logic in Computer Science (LICS'96), New Brunswick, NJ, USA, July 1996*, pages 313–321, 1996.

[AJ93] P. A. Abdulla and B. Jonsson. Verifying programs with unreliable channels. In *Proc. 8th IEEE Symp. Logic in Computer Science (LICS'93), Montreal, Canada, June 1993*, pages 160–170, 1993.

[AJ94] P. A. Abdulla and B. Jonsson. Undecidability of verifying programs with unreliable channels. In *Proc. 21st Int. Coll. Automata, Languages, and Programming (ICALP'94), Jerusalem, Israel, July 1994*, volume 820 of *Lecture Notes in Computer Science*, pages 316–327. Springer-Verlag, 1994.

[AJ97] P. A. Abdulla and B. Jonsson. Model-checking through constraint solving. In *Methods and Tools for the Verification of Infinite State Systems, Proceedings of the Grenoble-Alpe d'Huez European School of Computer Science, May 23–25, Grenoble, France.* VERIMAG, St-Martin d'Hères, France, 1997.

[BBK87] J. C. M. Baeten, J. A. Bergstra, and J. W. Klop. Decidability of bisimulation equivalence for processes generating context-free languages. In *Proc. Parallel Architectures and Languages Europe (PARLE'87), Eindhoven, NL, June 1987, vol. II: Parallel Languages*, volume 259 of *Lecture Notes in Computer Science*, pages 94–111. Springer-Verlag, 1987.

[BCG88] M. C. Browne, E. M. Clarke, and O. Grümberg. Characterizing finite Kripke structures in propositional temporal logic. *Theoretical Computer Science*, 59(1–2):115–131, 1988.

[BCM+92] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, 1992.

[Boc78] G. von Bochmann. Finite state description of communication protocols. *Computer Networks and ISDN Systems*, 2:361–372, 1978.

[Bon94] A. Bonchatbonrat. Concise Holmesian proofs for elementary Moncher-Watson problems. In A. Mycroft, editor, *Proc. 3rd Int. Conf. Theor. Crim. Deduction, London*, November 1894.

[Céc93] G. Cécé. *Etat de l'art des techniques d'analyse des automates finis communicants.* Rapport de DEA, Université de Paris-Sud, Orsay, France, September 1993.

[Čer94] K. Čerāns. Deciding properties of integral relational automata. In *Proc. 21st Int. Coll. Automata, Languages, and Programming (ICALP'94), Jerusalem, Israel, July 1994*, volume 820 of *Lecture Notes in Computer Science*, pages 35–46. Springer-Verlag, 1994.

[CFP95] G. Cécé, A. Finkel, and S. Purushothaman Iyer. Unreliable channels are easier to verify than perfect channels. *Information and Computation*, 124(1):20–31, 1995.

[Chr93] S. Christensen. Decidability and decomposition in process algebras. PhD thesis CST-105-93, Dept. of Computer Science, University of Edinburgh, UK, 1993.

[Cia94] G. Ciardo. Petri nets with marking-dependent arc cardinality: Properties and analysis. In *Proc. 15th Int. Conf. Applications and Theory of Petri Nets, Zaragoza, Spain, June 1994*, volume 815 of *Lecture Notes in Computer Science*, pages 179–198. Springer-Verlag, 1994.

[Esp97] J. Esparza. More infinite results. In *Proc. 1st Int. Workshop on Verification of Infinite State Systems (INFINITY'96), Pisa, Italy, Aug. 1996*, volume 5 of *Electronic Notes in Theor. Comp. Sci.* Elsevier, 1997.

[FC87] A. Finkel and A. Choquet. Fifo nets without order deadlock. *Acta Informatica*, 25(1):15–36, 1987.

[Fin82] A. Finkel. About monogeneous fifo Petri nets. In *Proc. 3rd European Workshop on Applications and Theory of Petri Nets, Varenna, Italy, Sep. 1982*, pages 175–192, 1982.

[Fin87a] A. Finkel. A generalization of the procedure of Karp and Miller to well structured transition systems. In *Proc. 14th Int. Coll. Automata, Languages, and Programming (ICALP'87), Karlsruhe, FRG, July 1987*, volume 267 of *Lecture Notes in Computer Science*, pages 499–508. Springer-Verlag, 1987.

[Fin87b] A. Finkel. Well structured transition systems. Research Report 365, Lab. de Recherche en Informatique (LRI), Univ. Paris-Sud, Orsay, August 1987.

[Fin90] A. Finkel. Reduction and covering of infinite reachability trees. *Information and Computation*, 89(2):144–179, 1990.

[Fin93] A. Finkel. The minimal coverability graph algorithm. In *Advances in Petri Nets 1993*, volume 674 of *Lecture Notes in Computer Science*, pages 210–243. Springer-Verlag, 1993.

[Fin94] A. Finkel. Decidability of the termination problem for completely specificied protocols. *Distributed Computing*, 7:129–135, 1994.

[GR85] M. G. Gouda and L. E. Rosier. Synchronizable networks of communicating finite state machines. Unpublished manuscript, 1985.

[HM85] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1):137–161, 1985.

[KS97] O. Kouchnarenko and Ph. Schnoebelen. A model for recursive-parallel programs. In *Proc. 1st Int. Workshop on Verification of Infinite State Systems (INFINITY'96), Pisa, Italy, Aug. 1996*, volume 5 of *Electronic Notes in Theor. Comp. Sci.* Elsevier, 1997.

[LC94] C. Lakos and S. Christensen. A general approach to arc extensions for coloured Petri nets. In *Proc. 15th Int. Conf. Applications and Theory of Petri Nets, Zaragoza, Spain, June 1994*, volume 815 of *Lecture Notes in Computer Science*, pages 338–357. Springer-Verlag, 1994.

[Mäk85] E. Mäkinen. On permutative grammars generating context-free languages. *BIT*, 25:604–610, 1985.

[Mol96] F. Moller. Infinite results. In *Proc. 7th Int. Conf. Concurrency Theory (CONCUR'96), Pisa, Italy, Aug. 1996*, volume 1119 of *Lecture Notes in Computer Science*, pages 195–216. Springer-Verlag, 1996.

[Val78] R. Valk. Self-modifying nets, a natural extension of Petri nets. In *Proc. 5th Int. Coll. Automata, Languages, and Programming (ICALP'78), Udine, Italy, Jul. 1978*, volume 62 of *Lecture Notes in Computer Science*, pages 464–476. Springer-Verlag, 1978.

[VJ85] R. Valk and M. Jantzen. The residue of vector sets with applications to decidability problems in Petri nets. *Acta Informatica*, 21:643–674, 1985.