Automated Generation of a Progress Measure for the Sweep-Line Method

Karsten Schmidt

Humboldt–Universität zu Berlin
Institut für Informatik
D–10099 Berlin

Abstract. In the context of Petri nets, we propose an automated construction of a *progress measure* which is an important pre-requisite for a state space reduction technique called the *sweep-line method*. Our construction is based on linear-algebraic considerations concerning the transition vectors of the Petri net under consideration.

1 Introduction

The sweep-line method [Mai03] is a recently proposed reduction technique for explicit state space verification. In its basic shape, it deletes previously constructed states that cannot serve as successors of states not yet explored. The key concept for this method is a so-called *progress measure* that assigns values to states which are non-decreasing w.r.t. the successor state relation. The sweep-line method was later generalized such that progress measures can be used which are non-monotonous w.r.t. the successor relation. In that case, states that have a predecessor with larger progress value are stored permanently. Thus, a good non-monotonous progress measure should be designed such that value decrease by transition occurrence happens as seldom as possible. In the original papers [CKM01,KM02], it is left to the user to provide a progress measure, assuming that the user knows about some concept of progress in the modeled system.

We propose an automated generation of a progress measure for the generalized sweep-line method. It works for place/transition Petri nets, where convenient concepts for describing progress measures cannot be found within the formalism itself (in contrast to high level nets where the language of annotations to the net can be used to formulate progress measures).

Our progress measure is not necessarily monotonous. We derive the measure from an analysis of the system's transition vectors, and their linear dependencies. We arrive at an incremental progress measure. That is, we can assign to each transition a fixed value such that the progress value of a successor state differs from the original state exactly by the value assigned to the fired transition. One advantage of this kind of progress measure is that, upon a transition occurrence, the progress value of the successor state can be computed by addition of an offset to the progress value of the predecessor, that is, in constant time. Moreover, so-called regress transitions—transitions that decrease the progress value—can be identified statically.

K. Jensen and A. Podelski (Eds.): TACAS 2004, LNCS 2988, pp. 192–204, 2004.

We start with a brief description of the sweep-line method, and continue with some basics about the linear algebra of Petri nets. Then, we present our proposal to the definition of a progress measure. Finally, we propose solutions concerning the combination of the sweep-line method with well-known state space reduction techniques such as partial order reduction or symmetry reduction.

2 The Sweep-Line Method

First, we sketch the basic sweep-line method [CKM01]. At any snapshot during explicit state space exploration, we can distinguish three kinds of states. We have states already seen, and states not yet seen. Among the states already seen there are those where all enabled transitions have already been explored, and those where some successors have not yet been explored. The last kind of states is called the *front*.

Assume we assigned a progress value[1] $p(s)$ to each state s such that for all transitions t, $s \xrightarrow{t} s'$ implies $p(s) \leq p(s')$. Obviously, all states still to be tested for presence in the state space are (transitive) successors of states in the front and have thus a progress value greater or equal to the minimum progress value among the front states. Consequently, states with smaller progress values than the minimum progress value appearing in the front can be safely removed from the state space. This is exactly the reduction the sweep-line method aims at. As the front evolves forward, more and more states can be removed, leading to the intuition of a sweep-line following the front of state space exploration and removing every state behind it (cf. Fig. 1). For being able to remove as many states as possible, and as early as possible, a search strategy is recommendable where front states are explored in ascending order of their progress values. In contrast, depth first search is not recommendable as the initial state is the last one to leave the front thus making it impossible for the sweep-line to proceed forward.

For the generalized sweep line method [KM02], the monotony condition for the progress measure is dropped. Thus, the method works, at least in principle, with any assignment p of progress values to states. Now, there can be situations where a transition leads to a state with smaller progress value. Such a pair of states is called a *regress edge* in the state space.

The generalized sweep-line method complements the basic method with the following twist. Whenever a regress edge occurs during a run (as in the basic method), the target state of that edge (the state with smaller progress value) is stored and marked *persistent*. That is, it will never be removed subsequently. It is, however, not explored immediately. Due to the removal of states behind the sweep line, we cannot be sure whether or not we have already seen that state. Thus, after having finished one state space exploration, we start another state

[1] in general, progress values can be members of any partially ordered set. For this paper, however, it is sufficient to view progress values as integer numbers. p is not necessarily injective.

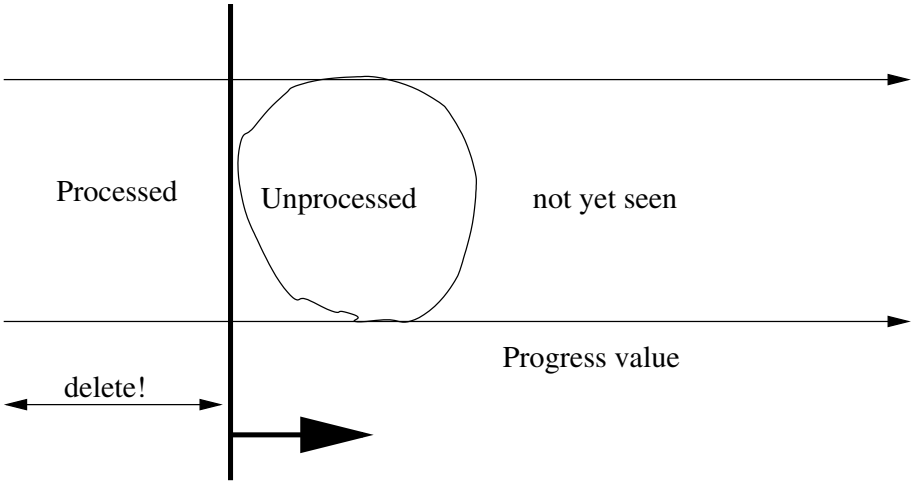

Fig. 1. A snapshot during sweep-line state space generation. States fully explored (processed) have smaller progress values than states not yet completely explored (unprocessed) and states not yet seen. States behind the sweep-line (following the unprocessed states) can be deleted.

space exploration with all states recently marked persistent as initial front. This exploration possibly re-explores parts of the state space, and can lead to further persistent states that need to be explored subsequently. It can, however, be shown that, for a finite-state system, every reachable state is visited at least once, so simple reachability queries can be verified using the method. Furthermore, the number of iterations until no additional persistent states are discovered, tends to be small.

3 Definitions

We use the notation $[P, T, F, W, m_0]$ for Petri nets, with the two finite and disjoint sets P (places) and T (transitions), the relation $F \subseteq (P \times T) \cup (T \times P)$ (arcs), the assignment $W : F \longrightarrow \mathbf{N} \setminus \{0\}$ (arc weights) and the initial marking m_0, where a marking is a mapping $m : P \longrightarrow \mathbf{N} \cup \{0\}$.

We extend W to $(P \times T) \cup (T \times P)$ by setting $W(x, y) = 0$ for $[x, y] \notin F$.

For a transition t, place vector Δt is defined by $\Delta t(p) = W(t, p) - W(p, t)$. Transition t is enabled at a marking m iff, for all $p \in P$, $m(p) \geq W(p, t)$. If t is enabled at m, t can fire at m leading to the successor state $m' = m + \Delta t$ (notation: $m \xrightarrow{t} m'$). The reachability relation $\xrightarrow{t}$ is extended to transition sequences in the canonic way, $m \xrightarrow{*} m'$ denotes reachability of m' from m by any finite transition sequence.

The incidence matrix C is a matrix with P as row index set and T as column index set, where for all transitions t, the corresponding column in C is equal to

Δt. A transition invariant is an integer solution to the system $C \cdot x = \underline{0}$ where x is a transition indexed vector of unknowns, and $\underline{0}$ is the place indexed vector of zeros.

Let $m \xrightarrow{t_1 \ldots t_n} m'$. By definition, we have $m' = m + \Delta t_1 + \cdots + \Delta t_n$. This equation can be rewritten to $m' = m + C \cdot \Psi(t_1 \ldots t_n)$ with Ψ being a vector with index set T where the entry for t is equal to the number of occurrences of t in $t_1 \ldots t_n$ (in the sequel called the *count vector* of the sequence). Equation $m' = m + C \cdot \Psi(t_1 \ldots t_n)$ is called the *state equation* for Petri nets.

A vector v is linear dependent on a set $\{v_1, \ldots, v_n\}$ of vectors if there are (rational) numbers $\lambda_1, \ldots, \lambda_n$ such that $v = \lambda_1 \cdot v_1 + \cdots + \lambda_n \cdot v_n$. A set of vectors is linear independent iff none of its members is linear dependent on the set of remaining members. For a matrix C, its rank $r(C)$ is defined as the size of the largest set of linear independent columns of C.

4 Progress Measures

A progress measure is a mapping $p : \mathbf{N}^P \longrightarrow A$, where A is an arbitrary set with a partial order $\leq$. If, for markings m, m', and a transition t, $m \xrightarrow{t} m'$ and $p(m) \not\leq p(m')$, $[m.m']$ is called a *regress edge*. A progress measure is monotonous if there are no regress edges between any two reachable markings.

Our progress measures map into the set $\mathbf{Q}$ of rational numbers, with the usual $\leq$ as its partial order. In addition, they are incremental.

A progress measure p is *incremental* if, for each $t \in T$, there is a rational number $o(t)$ (t's *offset*) such that for all m, m', $m \xrightarrow{t} m'$ implies $p(m') = p(m) + o(t)$. An incremental measure is uniquely defined by its transition offsets and the progress value of the initial marking. The progress value of the initial marking, however, does not influence the nature of the measure, so we assume $p(m_0) = 0$. Incremental progress measures have the advantage that calculation of progress values during depth first state space exploration can be done in constant time (just add the offset of the fired transition). Non-incremental measures may or may not require constant time calculations, depending on how many components of a state are considered for establishing progress.

Consider a marking m that is reached from the initial marking with two different transition sequences: $m_0 \xrightarrow{t_1 \ldots t_n} m$ and $m_0 \xrightarrow{t'_1 \ldots t'_k} m$. A consistent incremental progress measure must assign a unique progress value to m. That is, the transition offsets need to satisfy: $\sum_{i=1}^{n} o(t_i) = \sum_{j=1}^{k} o(t'_j)$.

Since the calculation of a progress measure needs to be performed prior to the state space generation, we do not have sufficient information about executable transition sequences. By the state equations $m = m_0 + C \cdot \Psi(t_1 \ldots t_n)$ and $m = m_0 + C \cdot \Psi(t'_1 \ldots t'_m)$ we know, however, that at least one of the transitions $t_1, \ldots, t_n, t'_1, \ldots, t'_k$ is linear dependent on the remaining transitions in this list. For a consistent measure, it is thus important to assign compatible values as soon as a transition is linear dependent on other ones.

Our approach works as follows. First, we determine a maximum size set U of linear independent transitions (the size of U is thus $r(C)$). This can be done

in polynomial time. For each $t \in U$ we set $o(t) = 1$. Since U is of maximum size, all remaining transitions can be expressed (uniquely!) as linear combinations of transitions in U. These linear combinations determine the remaining offsets as follows. Let $U = \{t_1, ..., t_n\}$ and $t \notin U$. Then there exist $\lambda_1, \ldots, \lambda_n$ such that $\Delta t = \lambda_1 \Delta t_1 + \cdots + \lambda_n \Delta t_n$. We set $o(t) = \lambda_1 o(t_1) + \ldots \lambda_n o(t_n)$ $(= \lambda_1 + \ldots \lambda_n)$. This value can be greater than, equal to, or less than 0 (Fig. 2). Thus, among the transitions outside U, there may be regress transitions. It is obvious that our setting leads to a consistent progress measure.

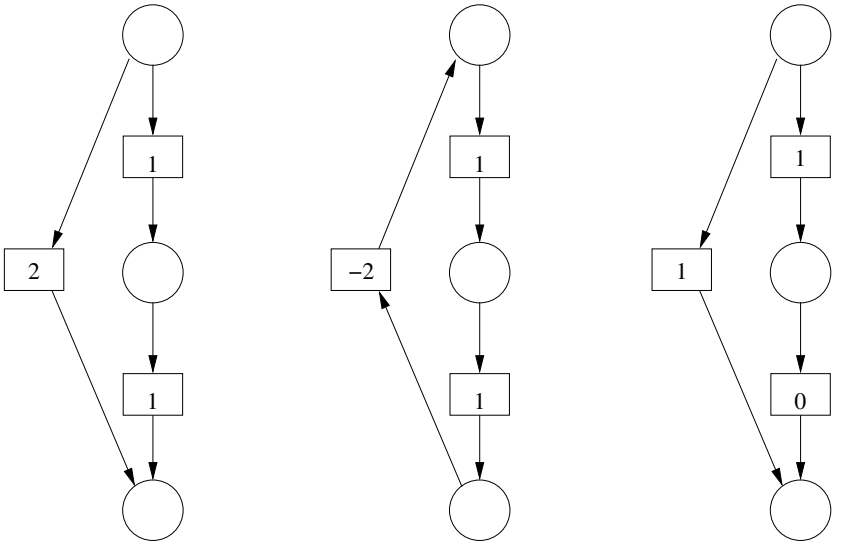

Fig. 2. The numbers in this figure are offsets. The transitions with offset 1 are members of U. The offset of the remaining transitions is determined by their linear combination from U.

Figure 3 contains a geometric interpretation of the just defined progress measure. Consider the euclidian space $Q^{|P|}$ of points with rational coordinates. Every marking defines a point in this space (with integer, even natural numbers as coordinates). The transition vectors Δt can be viewed as vectors. If $m \xrightarrow{t} m'$ then point m' is the translation of point m by vector Δt.

Linear independent vectors define a hyperplane E (the minimum size plane that contains the points defined by the translation of the point $\underline{0}$ by the respective vectors). E does not contain $\underline{0}$, so there is a unique point d in E that has minimal distance to $\underline{0}$ w.r.t. the usual euclidian metric. $\underline{0}$ and d define a line g containing both points. The progress value of a marking (a point) m is the distance of the unique intersection point i between line g and the parallel of E containing m, measured in a scale where the distance between $\underline{0}$ and d defines the unit (value 1).

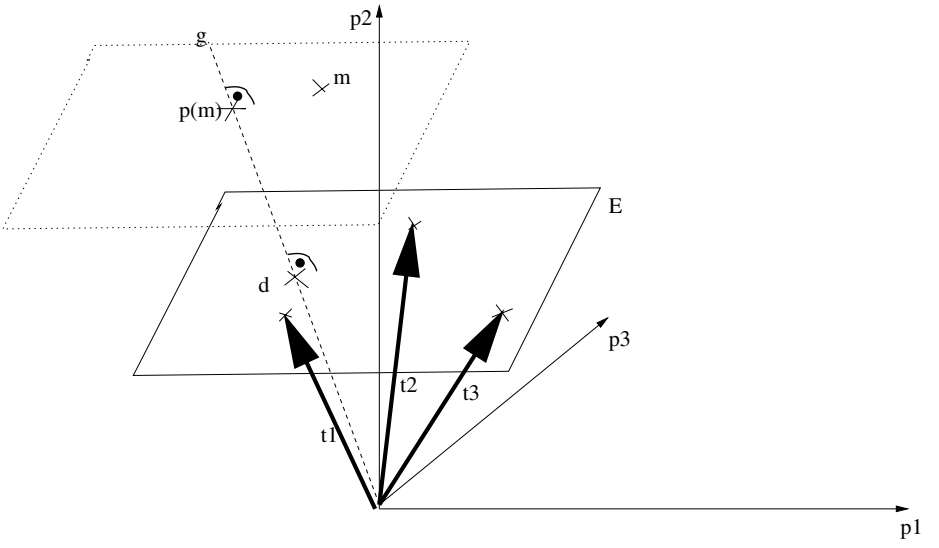

Fig. 3. Geometric interpretation of our progress measure

5 Possible Optimizations

The choice to assign offset 1 to the transitions in U is somewhat arbitrary. In fact, any assignment of offsets to transitions in U can be extended to a consistent progress measure for the whole system, following the linear combinations approach.

By changing offsets of transitions in U, the position of the plane E (and thus the direction of the progress line g) in Fig. 3 can be controlled. An optimal position would be such that as many as possible transitions point to the same side of the parallel of E through point $\underline{0}$. We tried to formulate this as an optimization problem, but did not succeed to arrive at a linear one. However, standard optimization heuristics such as simulated annealing or hill climbing may help to find close to optimal solutions. We cannot quantify the effect of such techniques at this time.

Approximating the target function (trying, for instance, to maximize the *sum* of offsets of all transitions rather than the number of transitions with positive offsets), leads to unsatisfactory results. For the sweep-line method, it is better to have many transitions with small positive offset, and few transitions with large negative offset than to have any kind of more balanced assignment.

Another source of possible optimization is the actual choice of U. While the size of U is determined by the rank of the incidence matrix C, there is some freedom left to choose its actual constituents. It might be possible to control the calculation of U such that, for instance, regress transitions do not form long chains (which would possibly lead to a large number of iterations of the sweep-line method). Results in this regard must as well be left to future work.

6 Combination with Other State Space Reduction Techniques

We consider first partial order reduction [Pel93,GW91,Val88]. Since the sweep-line method is able to investigate the set of reachable states, but not the mutual connections between states, only simple safety properties can be verified. Among them are presence of deadlocks [Val88], existence of dead transitions [Val91, Sch99b], reachability [Val91,Val93,Sch99b,KV00], and similar simple safety properties [Val93]. See [Val96,Val98] for surveys on partial order reduction methods. Partial order reduction consists of computing, in every state, a subset of the enabled transitions such that the considered property is preserved when only the transitions in the computed subset are fired. In most of the cited techniques, the subset can be determined by examining the structure of the system (here: the Petri net) and the current state. This is particularly the case for the deadlock preserving method [Val91], the methods proposed in [Sch99b] for dead transitions and reachability, and the approach in [Val93].

In [Val91], a core concept is a solution to the so-called ignorance problem. It requires detection of cycles or strongly connected components in order to avoid infinite ignorance of certain transitions. The approach of [KV00] to reachability has a similar condition. Traditionally, graph structures such as cycles or strongly connected components are detected through extensions to the depth-first search algorithm [Tar72]. Since depth first search is not available with the sweep-line method (as the initial state would be kept in the unprocessed set until the very end of state space exploration), we need a different implementation. Fortunately, a progress measure gives sufficient information, at least about cycles: every cycle contains a regress transitions (one with negative offset), or all transitions in the cycle have offset 0. While regress transitions can be immediately discovered during state space exploration, cycles containing only 0 offset transitions can as well be discovered since the involved states share the same progress value and are only deleted after having explored all of them. This means: if a property is preserved by the sweep-line method, then all existing partial order reduction approaches for that property can be combined with the sweep-line method, though not necessarily in their most efficient fashion.

Next, we consider the symmetry method [HJJJ84,CEFJ96,ES96,ID96, Sch00a,Sch00b]. Its principal compatibility to the sweep-line method was already mentioned in [Mai03]. However, there is a serious problem with the particular approach proposed in this paper. The symmetry method works by defining a suitable equivalence on the set of states, and exploring only one representative of every equivalence class. The usual implementation requires transforming every freshly computed state into a canonical representative of its class. The problem with our approach to the sweep-line method is that , ad 1, the canonical representative does not necessarily have the same progress value as the original state, and ad 2, the (incrementally determined) progress value of the canonical representative cannot be determined without having an actual transition sequence to

this state. The only suitable solution to the problem is to arrange the progress measure such that equivalent states receive equal progress values.

In [Sch00a], we detect symmetries as graph automorphisms of the underlying Petri net. An automorphism is a bijection on the set of nodes (places and transitions) that respects node type, neighborhood w.r.t. the arc relation, multiplicities, and initial marking. The group of graph automorphisms defines an equivalence relation on the nodes as well as on the states of the Petri net. Two nodes are equivalent if there is a graph automorphism that maps one of them onto the other. Two states are equivalent if there is a graph automorphism that maps one of them onto the other. Thereby, for a state m and a graph automorphism σ, $\sigma(m)$ is defined by the equations $\sigma(m)(\sigma(p)) = m(p)$, for all places p. In this approach it can be proven that, if a state is reached by a sequence of transitions, equivalent states can also be reached by a sequence of equivalent transitions. Formally, $m_1 \xrightarrow{t_1 t_2 \ldots t_{n-1}} m_n$ implies $\sigma(m_1) \xrightarrow{\sigma(t_1)\sigma(t_2)\ldots\sigma(t_{n-1})} \sigma(m_n)$, for all graph automorphisms σ. Since every automorphism is expected to respect the initial state m_0, we have $\sigma(m_0) = m_0$, for all automorphisms σ.

Assume an incremental progress measure p defined by offsets $o(t_1), \ldots, o(t_n)$ to transitions $t_1, \ldots, t_n$, Let m be a state reachable from the initial state through a sequence $t_1 \ldots t_n$. Then, $p(m) = p(m_0) + o(t_1) + \cdots + o(t_n)$. For any state m' equivalent to m, we have an automorphism σ such that $m' = \sigma(m)$. Consequently, we have $p(m') = p(m_0) + o(\sigma(t_1)) + \cdots + o(\sigma(t_n))$. For achieving a symmetry respecting progress measure, it is thus sufficient to assign offsets to transitions such that equivalent transitions receive equal offsets. Then, $o(t_i) = o(\sigma(t_i))$ for all σ and all i. Unfortunately, this idea cannot be easily integrated into the approach presented so far, since the set U is not necessarily closed under the equivalence relation induced by symmetry. Furthermore, it can happen that transitions outside U, though equivalent, receive different values even if equivalent transitions inside U have equal values. It is thus necessary to compute a symmetry respecting progress measure with a more complicated approach. In the first step, we compute a generating set of all transition invariants of the Petri net, i.e. a set of invariants such that every transition invariant is a linear combination of these ones. Since we consider rational solutions, the computation can be done in polynomial time. In a second step, we calculate the offsets $(o(t_1), ..., o(t_n))$ of transitions $t_1, ..., t_n$ (where $n = card(T)$) as solutions of a homogeneous system of equations, including the following equations. First, for every generator $(a_1, ..., a_n)$ of the set of transition invariants, add equation $a_1 o(t_1) + \cdots + a_n o(t_n) = 0$. These equations guarantee consistency of the computed measure. Second, we add, for every pair $[t_i, t_j]$ of equivalent transitions, the equation $o(t_i) = o(t_j)$. Thus, solutions become symmetry respecting. Using a simple solution method such as Gaussian elimination, we may eventually choose some of the offset values arbitrarily. We use this opportunity to assign a positive value at least to those transitions. Every such assignment induces a positive value to a whole equivalence class (through the second kind of equations). This approach is more complicated since it requires the subsequent solution of two systems of equations. In terms of the computed progress measure, it tends,

however, to yield acceptable measures, even though there exist cases where a non-trivial progress measure exists, but no non-trivial symmetry respecting progress measure (see Fig. 4).

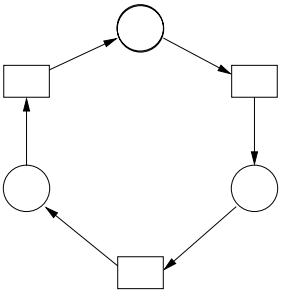

Fig. 4. This net has a progress measure where two of the transitions have offset 1 while the remaining transition has offset -2. All transitions are equivalent w.r.t. symmetry. The only symmetry respecting progress measure is thus the one assigning 0 to all transitions, and no reduction can be achieved.

Consistency of the proposed symmetry respecting progress measure can be verified as follows. Assume, one and the same state m is reached from the initial state through different transition sequences $t_1 \dots t_k$ and $t'_1 \dots t'_m$. The Petri net state equation yields $m = m_0 + C \cdot \Psi(t_1 \dots t_k)$ and $m = m_0 + \Psi(t'_1 \dots t'_m)$. Thus, $C \cdot (\Psi(t_1 \dots t_k) - \Psi(t'_1 \dots t'_m)) = 0$ which means that $\Psi(t_1 \dots t_k) - \Psi(t'_1 \dots t'_m)$ is a transition invariant. Since, through the first kind of equations introduced above, every *generator* $(a_1, \dots, a_n)$ of the set of transition invariants satisfies $a_1 \cdot o(t_1) + \dots a_n \cdot o(t_n) = 0$. Since every transition invariant can be expressed as a linear combination of the generators, we have that actually *all* transition invariants $(b_1, \dots, b_n)$ satisfy $b_1 \cdot o(t_1) + \dots b_n \cdot o(t_n)) = 0$. In particular, we may conclude for the above mentioned difference of count vectors, $o(t_1) + \dots o(t_k) - (o(t'_1) + \dots o(t'_m)) = 0$, or, equivalently, $o(t_1) + \dots o(t_k) = o(t'_1) + \dots o(t'_m)$. m thus receives the same progress value, no matter which sequence is used to compute it. This means that the measure is consistent.

7 Examples

We consider first the system of dining philosophers, Fig. 5. This is a system consisting of a large number of agents where each agent performs a repeated sequence of only few actions. Thus, the state space of this system has a huge number of cycles where many of them can be performed independently. Consequently, most of the states of the system are reached by at least one regress transition (there is exactly one regress transition per philosopher). This nature of the dining philosophers system is actually an indication for *not* applying the sweep-line method which is confirmed by the results below (see the number of

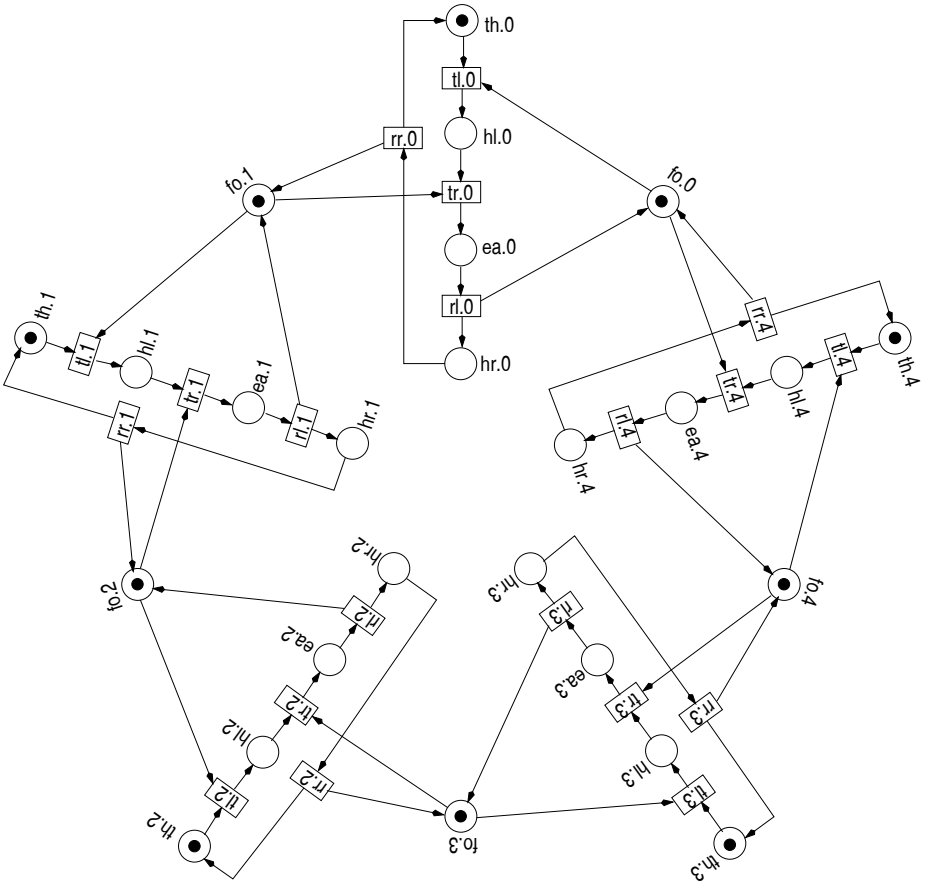

Fig. 5. The five dining philosophers system. A philosopher acts in the cycle take left fork - take right fork - release left fork - release right fork.

persistent states). We have included this system in order to show that, even for systems with no immediate concept of global progress, the sweep-line method *is* able to reduce peak memory usage. It is only necessary to combine it with partial order reduction. Partial order reduction has the effect of reducing the number of interleavings. Thus, local cycles of independent (not neighbored) philosophers are decoupled and the ratio between persistent and non-persistent states is much smaller. For realistic reactive systems, we expect an even better reduction since local cycles tend to be larger than in academic examples thus decreasing the number of regress transitions. The progress measure computed by our approach has three transitions per philosopher with offset 1, and the remaining transition with offset -3. Declaring every but one local step of a philosopher a "progress", and the remaining one as regress, appears natural even for a human generated progress measure. A person would, however, choose the same local transition to

be the regress one (thus making the measure symmetry respecting) while our implementation does not compute a symmetry respecting solution.

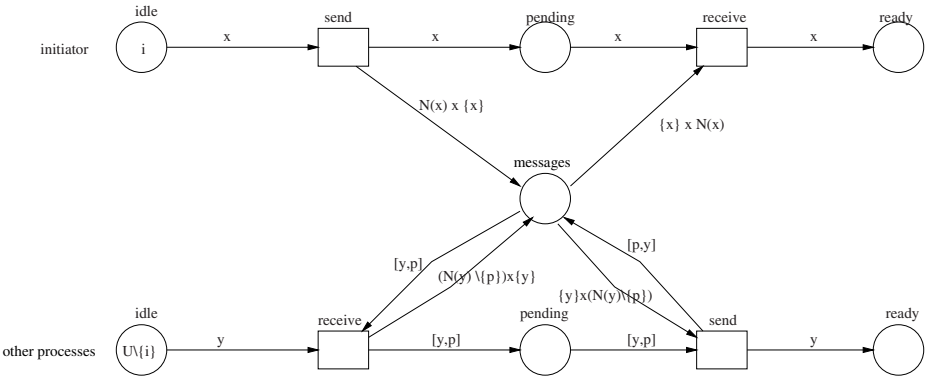

Fig. 6. The ECHO broadcasting protocol depicted as a high level net. The initiator (on top) sends messages to all its neighbors, and waits for corresponding acknowledgments. Other agents (on bottom) send (on receipt of one message), messages to all remaining neighbors, and send, after having collected corresponding acknowledgments, and acknowledgment to the originator of the message.

The second example, Fig. 6, shows a distributed algorithm for propagating information with feedback. The algorithm terminates. All transitions are linear independent. We are therefore able to compute a monotonous progress measure assigning 1 as offset to all transitions. This coincides with the human intuition that in a terminating algorithm, every executed step is "progress". We can see that in acyclic systems, the sweep-line method performs well, even if applied in isolation.

The data reported in Table 1 have been collected using the tool LoLA [Sch99a]. The implementation of the sweep-line method in LoLA is complete, including the fully automated determination of a progress measure according to the reported approach. It will be publicly available (open source) with the next release of the tool.

The numbers concerning partial order reduction concern deadlock preserving stubborn sets, but without considering the impact of on-the-fly verification. That is, state space exploration continues even after deadlocks have been found. Otherwise, numbers would be such small that no real comparison could be done.

The examples, as a few other experiments, suggest that the automatically computed progress measures are competitive to user-defined measure.

8 Conclusion

We have shown that an automated construction of a progress measure, by considering linear dependencies between transitions, leads to well-performing progress

Table 1. Experimental results: PHi = i dining philosophers, ECHOi = broadcasting algorithm for i agents in a grid-like network. Empty field = experiment not conducted; ? = out of memory. Platform: LINUX on 650 MHz Intel processor, 378 MB RAM

	PH5	PH10	PH12	PH100	PH400	ECHO3	ECHO9	ECHO25
full state space								
states	242	59048	531440	$3^{100}-1$	$3^{400}-1$	11	2628	?
trans. fired	805	393650	4251516	?	?	14	9994	?
time (sec)	0.1	5.8	71.0	?	?	0.0	0.5	?
sweep-line method								
nr. of iterations	3	3	3	?	?	1	1	1
peak nr. of states	183	54122	502378	?	?	4	634	260564
nr. persistent states	169	53299	497969	?	?	0	0	0
trans. fired	1544	720428	7710664	?	?	14	6805	1311085
time (sec)	0.1	24.3	277.8	?	?	0.1	0.1	284.7
partial order reduced state space								
states		272		29702	478802	7	1433	?
trans. fired		370		39700	638800	6	2463	?
time (sec)		0.5		2.3	115.3	0.0	0.2	?
sweep-line method plus partial order reduction								
nr. of iterations		3		11	41	1	1	1
peak nr. of states		97		9141	144936	2	340	236941
nr. persistent states		75		8929	144117	0	0	0
trans. fired		700		75800	1215204	6	2267	870495
time (sec)		0.1		17.2	1712.0	0.1	0.3	193.7

measures. For the examples discussed in the previous section, it is hard to imagine any user-defined measures that would perform significantly better. It has also become evident that a combination between the sweep-line method and partial order reduction is highly recommendable in the case of reactive systems. In the proposed fashion, the sweep-line method is well suited for the verification of low level Petri net models.

For high level models, our method can be applied in two ways. We can either unfold the high level net to a low level net and apply our method as such, or we can perform dependency analysis on the skeleton of the colored net instead. If the inscriptions of the high level net produce and consume a number of tokens that does not depend on particular transition bindings, this would lead to a sound progress measure. In the ECHO example, applied to a homogeneous network (one where all agents have the same number of neighbors), this method could be applied and would also result in a monotonous progress measure. It should be mentioned, though, that progress in high level nets does sometimes occur as a monotonous evolution of data values on certain tokens. In such cases, user defined progress measures are, without doubt, superior to automatically computed progress measures.

References

[CEFJ96] E.M. Clarke, R. Enders, T. Filkorn, and S. Jha. Exploiting symmetry in temporal logic model checking. *Formal Methods in System Design 9*, pages 77–104, 1996.

[CKM01] S. Christensen, L.M. Kristensen, and T. Mailund. A sweep-line method for state space exploration. *Proc. TACAS 01, LNCS*, 2031:450–464, 2001.

[ES96] E.A. Emerson and A.P. Sistla. Symmetry and model checking. *Formal Methods in System Design 9*, pages 105–131, 1996.

[GW91] P. Godefroid and P. Wolper. A partial approach to model checking. *6th IEEE Symp. on Logic in Computer Science, Amsterdam*, pages 406–415, 1991.

[HJJJ84] Huber, A. Jensen, Jepsen, and K. Jensen. Towards reachability trees for high–level petri nets. In *Advances in Petri Nets 1984, Lecture Notes on Computer Science 188*, pages 215–233, 1984.

[ID96] C.N. Ip and D.L. Dill. Better verification through symmetry. *Formal Methods in System Design 9*, pages 41–75, 1996.

[KM02] L.M. Kristensen and T. Mailund. A generalized sweep-line method for safety properties. *Proc. FME 02, LNCS*, 2391:549–567, 2002.

[KV00] L.M. Krisensen and A. Valmari. Improved question-guided stubborn set methods for state properties. *Proc. 21th Int. Conf. Application and Theory of Petri nets*, pages 282–302, 2000.

[Mai03] T. Mailund. *Sweeping the state space.* PhD thesis, Univerity of Aarhus, 2003.

[Pel93] D. Peled. All from one, one for all: on model–checking using representitives. *5th Int. Conf. Computer Aided Verification,Elounda, Greece, LNCS 697*, pages 409–423, 1993.

[Sch99a] K. Schmidt. Lola: A low level analyser. *Proc. Int. Conf. Application and Theory of Petri net, LNCS*, 1825:465–474, 1999.

[Sch99b] K. Schmidt. Stubborn set for standard properties. *Proc. 20th Int. Conf. Application and Theory of Petri nets, LNCS 1639*, pages 46–65, 1999.

[Sch00a] K. Schmidt. How to calculate symmetries of petri nets. *Acta Informatica 36,*, pages 545–590, 2000.

[Sch00b] K. Schmidt. Integrating low level symmetries into reachability analysis. *Proc. 6th Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems, LNCS 1785*, pages 315–331, 2000.

[Tar72] R. E. Tarjan. Depth first search and linear graph algorithms. *SIAM J. Comput.*, 1:146–160, 1972.

[Val88] A. Valmari. Error detection by reduced reachability graph generation. *Proc. of the 9th European Workshop on Application and Theory of Petri Nets, Venice*, 1988.

[Val91] A. Valmari. Stubborn sets for reduced state space generation. *Advances of Petri Nets 1990, LNCS 483*, pages 491–511, 1991.

[Val93] A. Valmari. On-the-fly verification with stubborn sets. *5th Int. Conf. Computer Aided Verification,Elounda, Greece, LNCS 697*, pages 397–408, 1993.

[Val96] A. Valmari. Stubborn set methods for process algebras. *Workshop on Partial Order Methods in Verification, Princeton*, pages 192–210, 1996.

[Val98] A. Valmari. The state explosion problem. *Lectures on Petri nets I: Basic models, LNCS 1491*, pages 429–528, 1998.