Model Checking in Span(Graph)

Robbie Gates[1], Piergiulio Katis[1], N. Sabadini[2], R.F.C.Walters[1]

[1] School of Mathematics and Statistics, University of Sydney, Australia,
[2] Dipartimento di Scienze dell'Informazione, Universitá di Milano,Italy

Katis, Sabadini and Walters have introduced in [2] an algebra for systems with boundaries, namely **Span(Graph)**. To an expression in the algebra there is an associated graphical representation, which corresponds to the distributed nature of the system. The original motivation for considering **Span(Graph)** in [2] was to have a compositional algebra of finite transition systems with operations relating to those of [1], [5], and Kurshan. It the purpose of this note to give a new application of this algebras, namely to model checking. Given a system as an expression in **Span(Graph)** the algebra allows the definition of notions such as subsystem, state internal to a subsystem, local independence of actions etc. Using these transparent compositional notions we are able to give a clear and intuitive account of model checking techniques and to give complexity results in a variety of cases. The existing techniques of partial order model checking, for example [V88], [V93], are on the contrary based on non-compositional models of concurrency and as a result the problem of finding local independence, stubborn sets, etc is difficult. The compositional structure is in fact available to designers of systems, but the information is ignored in these approaches.

The spans of reflexive graphs considered are finite, linear and come equipped with an initial vertex. Given an expression Ψ of such spans, we call an edge of (the head of the value of) Ψ *atomic* if it is one of the following types: reflexive in each component of Ψ; reflexive in all but one component of Ψ; or reflexive in all but two components of Ψ, these two non-reflexive edges agreeing as a non-reflexive action on a common boundary (in other words, the two edges engage in a non-trivial synchronisation). Define the graph $\tilde{\Psi}$ to be the subgraph of Ψ comprising the vertices reachable from the initial vertex of Ψ and the atomic edges between these vertices. (The initial vertex of Ψ comprises the initial vertices of each component of Ψ.) In analysing behaviours of Ψ the linearity of the compoents ensure that it is enough to consider behaviours in $\tilde{\Psi}$, that is behaviours comprising atomic edges, when searching for deadlock. We now describe methods for constructing subgraphs G of $\tilde{\Psi}$ such that (a) G contains the initial vertex and (b) every deadlock reachable in $\tilde{\Psi}$ (and hence in Ψ) is reachable in G. An *internal state* for a span is a vertex v such that v is not a deadlock and each edge with source v is labelled by reflexive edges on all of the boundaries. Given a subset S of the set of components used to construct Ψ there is an associated subexpression. We denote the set of sub-expressions of Ψ by $\mathcal{P}(\Psi)$. A *deadlock analysis* for an expression $\Psi : I \to I$ of spans is a function $F : \mathrm{Vert}(\Psi) \to \mathcal{P}(\Psi)$ such that, for each vertex v of Ψ, if $F(v)$ does not comprise all the components of Ψ then the restriction of v to $F(v)$ is an internal state for $F(v)$. Given a deadlock analysis F for Ψ, we can define a subgraph Ψ_F of $\tilde{\Psi}$ to be the smallest subgraph satisfying

the following specification: the subgraph Ψ_F contains the initial vertex; and if v is a vertex of Ψ_F then Ψ_F contains those edges e with source v such that, for each component G not in $F(v)$, the restriction of e to G is a reflexive edge. In other words, Ψ_F is constructed by starting at the initial state and then, at each vertex v, only considering motions in $F(v)$. Though Ψ_F is a subgraph of Ψ, it can be used to detect deadlock in the original system. In fact, we have the following result. *Suppose F is a deadlock analysis of Ψ. Then all the reachable deadlocks of Ψ are (reachable) in Ψ_F.* A deadlock analysis provides us with an algorithm for searching for deadlocks. When searching we only need to consider the subgraph Ψ_F – that is, motions in the sub-systems determined by the deadlock analysis. In some cases the deadlock analysis F can be chosen such that the number of vertices of the graph Ψ_F grows polynomially (rather than exponentially) with the number of components. We present a simple example as illustration. It is usually the case that to minimise the size of Ψ_F we should choose F such that each $F(v)$ contains as few components as possible. In some examples it happens that, for each v, there is a unique smallest $F(v)$ to choose. This is the case with the dining philosophers. Let $\Psi = \mathrm{Fb}((F \cdot P)^n)$ be the system of n dining philosophers and n forks. Now consider the subgraph $\tilde{\Psi}$ comprising the atomic edges; an edge of $\tilde{\Psi}$ corresponds to one fork being picked up or put down. Let v_{init} be the initial state corresponding to no philosopher having a fork. There is only one choice for $F(v_{\mathrm{init}})$, it must be the whole system. There are n edges with source v_{init}, each of these corresponding to one philosopher picking up his right fork. Suppose the vertex w is the target of one of these n edges; say that it corresponds to the i^{th} philosopher having his right fork. The smallest candidate for $F(w)$ is $\{P_{(i-1)(\mathrm{mod}\ n)}, F_{(i-1)(\mathrm{mod}\ n)}, P_i\}$. So, although there are many edges in the original system with source w, we need only consider two of these: one corresponds to the i^{th} philosopher taking his left (that is, second) fork; and the other corresponds to the $(i-1)(\mathrm{mod}\ n)^{\mathrm{th}}$ philosopher taking his right fork. If we continue to construct the deadlock analysis F in this way, we find that the number of vertices in Ψ_F is $3n^2 - 3n + 2$. (The reader should check that the unique reachable deadlock of this system – the vertex corresponding to each philosopher having his right fork – is indeed found by this method.) This quadratic polynomial is the one given in [4], which there describes the number of markings of a Petri net needed to detect deadlock using the stubborn set method.

References

1. A. Arnold, *Finite transition systems*, Prentice Hall, 1994.
2. Katis P, Sabadini N, Walters RFC, Span(Graph): an algebra of transition systems, Proceedings AMAST '97, SLNCS 1349, 322-336, 1997.
3. Valmari A., On-the-fly verification with stubborn sets, Proceedings CAV93, SLCS 697, 397-408, Springer Verlag, 1993.
4. Valmari A., Error detection by reduced reachability graph generation, Preprint, 1988.
5. Zielonka W., Note on asynchronous automata, RAIRO 1987.