Following references were used during the project on compilation of statecharts (SCOPE). I have studied most of them. Some I was not able to reach unfortunately.

Only some papers on the list directly discusses statechart-like languages and even fewer is concerned with the problem of code synthesis from model description. Hopefully I will be able to filter those to separate list once the project is over.

References


100. A. Knapp and S. Merz. Model checking and code generation for UML state machines and collaborations. In D. Haneberg, G. Schellhorn, and W. Reif,


On formal semantics of statecharts as supported by 
STATEMATE. In 2nd BCS-FACS Northern Formal 

133. E. Mikk, Y. Laknech, and M. Siegel. Hierarchical 
automata as model for statecharts. In R. K. Shyamasundar 
and K. Ueda, editors, Third Asian Computing Science Conference (ASIAN), volume 1345 of 
Lecture Notes in Computer Science, pages 181–196, 

134. E. Mikk, Y. Laknech, M. Siegel, and G. J. 
Holzmann. Implementing statecharts in 
PROMELA/SPIN. In 2nd IEEE Workshop on 
Industrial Strength Formal Specification Techniques, 


136. J. B. Moller. Symbolic Model Checking of Real-Time 
Systems Using Difference Decision Diagrams. PhD 
thesis, Department of Innovation, IT University of 
Copenhagen, Apr. 2002.

137. M. Murakami. Partial evaluation of reactive 
communicating processes using temporal logic formulas. 
In Workshop on Algebraic and Object-Oriented Approaches to Software Science, 1995.

138. I. A. Niaz and J. Tanaka. Code generation from 
UML statecharts. In 7th IASTED International 
Conference on Software Engineering and Applications (SEA 2003), Marina del Rey, CA, USA, Nov. 
2003.

139. J. Noble and C. Weir. Small Memory Software. Patterns 

140. Object Management Group. OMG Unified Modelling 

141. Object Management Group. Meta object facility 

142. Object Management Group. OMG XML metadata 

143. D. L. Parnas. On the design and development of 
program families. IEEE Transactions on Software 

144. A. Pnueli and M. Shalev. What is in a step: On the 
semantics of statecharts. In T. Ito and A. R. Meyer, 
editors, International Conference on Theoretical Aspects 
of Computer Science (TACS), volume 526 of 
Lecture Notes in Computer Science, pages 244– 
Springer-Verlag.

Validation Tool (CVT) – Automatic verification of 
code generated from synchronous languages. In The 
International Workshop on Software Tools for Techno-
BRICS.

146. S. K. Rajamani and J. Rehof. Conformance check-
ing for models of asynchronous message passing soft-
ware. In Brinksma and Larsen [33], pages 166–179.

147. S. Ramesh. Efficient translation of statecharts into 
hardware circuits. In 12th International Conference on 


149. M. Riebsch, I. Philippow, and M. Götte. UML-
based statistical test case generation. In M. Aksit, 
M. Mezini, and R. Unland, editors, Objects, Com-
ponents, Architectures, Services, and Applications 
for a Networked World. NetObjectDays, Revised Pa-

150. E. E. Roubtsova, J. van Katwijk, R. C. M. de Rooij, 
and H. Toetenel. Transformation of UML specification 
to XTG. In D. Björner, M. Broy, and A. V. 
Zamulin, editors, Perspectives of System Informatics (PSI), 4th International Andrei Ershov Memo-
rial Conference, Revised Papers, volume 2244 of Lecture 
Notes in Computer Science, pages 249–256, 

151. G. Rozenberg and A. Salomaa, editors. Handbook of 

152. A. Sangiovanni-Vincentelli and J. Sifakis, editors. 
2nd International Conference on Embedded Software 
Springer-Verlag.

153. S.A.Seshia, R. Shyamasundar, A. Bhattacharjee, and 
S. Dhodapkar. A translation of statecharts to 
Esterel. In J. M. Wing, J. Woodcock, and J. Davies, 
editors, FM’99 - Formal Methods, World Congress on Formal Methods in the Development of Comput-
ing Systems, volume 1709 of Lecture Notes in Com-
puter Science, pages 983–1007, Toulouse, France, 

154. Z. Sawa. Equivalence checking of non-netsystems is 
EXPTIME-hard. In FIXME, editor, Interna-
tional Conference on Concurrency Theory (CON-
CUR), Lecture Notes in Computer Science, page 
FIXME, Marseille, France, Sept. 2003. Springer-
Verlag.


156. K. Scott and N. Ramsey. When do match-
compilation heuristics matter? Technical Report


For code generation, YAKINDU Statechart Tools use a textual generator model called SGen. The generator model holds key parameters for the code generation process and allows for the latter’s customization. The first step to code generation is to create a new SGen model. Right-click on the model folder in the project explorer and select New → Code generator model from the context menu. Selecting New → Code generator model in the context menu. The YAKINDU generator model wizard opens. Change the File name to CallHandling.sgen, then click Next >. Selecting a filename for the generator model. Fr Statechart code. 25. Language-specific code generation features. After initial code generation for a configuration, code is generated only for elements that were modified since the last generation. This provides a significant performance improvement by reducing the time required for generating code. Rational Rhapsody provides a number of options that can be used to further improve code generation performance. Breaking code generation into chunks. When working with extremely large projects, you may encounter memory-related problems when trying to generate code for the entire model. To overcome such problems, you can use the [lang]_CG::Configuration::ClassesPerCGCall property to break up the code generation process into a number of distinct chunks.