Systematischer Entwurf, formale Modellierung und modellbasierte Validierung von Steuerungen in *Ablaufsprache*

Christian Sonntag, Stephan Fischer, Sebastian Engell

{christian.sonntag|stephan.fischer|sebastian.engell}@bci.tu-dortmund.de

Lehrstuhl für Systemdynamik und Prozessführung,
Fakultät Bio- und Chemieingenieurwesen, TU Dortmund

1 Einleitung

Ablaufsprache (*AS*, engl.: *Sequential Function Chart / SFC*) ist eine mächtige, in der Norm EN 61131-3 standardisierte graphische Programmiersprache, die sich als industrieller Standard für die Entwicklung von Ablaufsteuerungen etabliert hat. Dieser Beitrag stellt eine durchgängige Werkzeugkette vor (siehe Abb. 1), die den gesamten Entwicklungsprozess für Ablaufsteuerungen von der systematischen Verfeinerung informeller Spezifikationen in *AS*-Programme anhand der *DC/FT*-Methode [1,2] bis hin zu modellbasierten Validierung oder Optimierung abdeckt. Da die in der Norm EN 61131-3 definierten Ausführungsregeln für *AS*-Programme nicht eindeutig sind (siehe z.B. [3,4]), sodass augenscheinlich identische *AS*-Programme je nach Implementierung unterschiedlich ausgeführt werden können, basiert die Werkzeugkette auf einer formalen, eindeutigen Definition der Syntax und Semantik von *AS*, die zudem ausschließlich *sichere* AS-Programme erzeugt, die niemals in Deadlocks enden können [5,6]. Basierend auf dieser formalen Definition werden *AS*-Programme algorithmisch in automatenbasierte Modelle übersetzt, die zur modellbasierten Validierung (entweder durch Simulation oder durch rigorose Verifikation) und Optimierung an Anlagenmodelle in verschiedenen Modellformalismen gekoppelt werden können.

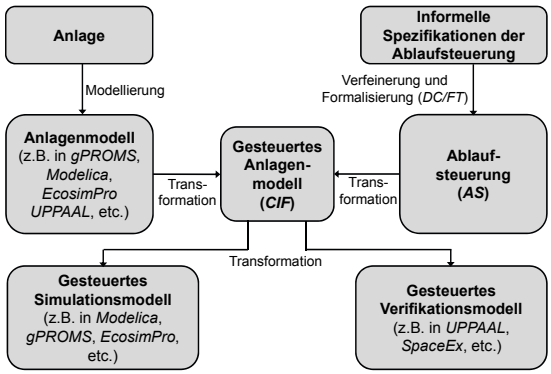

Abb. 1: Werkzeugkette für die Steuerungsentwicklung und –validierung.

2 Systematischer Entwurf von *AS*-Programmen

In der Praxis beginnt der Entwurf von Steuerungsprogrammen üblicherweise mit der Formulierung der Anforderungen. Diese sind meist in natürlicher Sprache formuliert, was eine Prüfung auf Vollständigkeit, Eindeutigkeit und Korrektheit erschwert. Die eigentliche Implementierung des Programms erfolgt anschließend manuell auf Basis dieser Spezifikation. Um diese fehleranfällige Prozedur zu verbessern, wurde die systematische *DC/FT*-Methodik (*Dependency Chart* / *Function Table*) entwickelt, um die informellen Anforderungen schrittweise zu formalisieren und anschließend automatisch in *AS*-Steuerungsprogramme zu übersetzen. Die Methode wurde in Form eines graphischen Softwarewerkzeugs realisiert [1,2].

3 Formale Modellierung von *Ablaufsprache*-Programmen

Die Werkzeugkette in diesem Beitrag basiert auf einer formalen, kompakten *AS*-Syntax, in der nur sichere *AS*-Programme erzeugt werden können [6]. Die Syntax eines *AS*-Programms wird durch nur vier Produktionsregeln definiert, die die Hauptelemente von *AS*-Programmen (siehe Abb. 2), also *Schritte* (mit assoziierten *Aktionen*), *Transitionen*, *Parallel-* und *Alternativverzweigungen*, sowie *Sprünge*, enthalten. Die Syntax ist eine Weiterentwicklung der Syntax in [4] bzw. [5]:

$$\psi_{SFC} ::= \ '\square' \ seq \tag{1}$$

$$seq ::= \ '\bot\square' \ seq \ |' \ \triangledown' \ alt \ ' \ \triangle \ \square' \ seq \ | \tag{2}$$
$$'\bot\blacktriangledown' \ par \ '\blacktriangle\bot\square' \ seq \ | \ \emptyset$$

$$alt ::= \ \{ \ '[\bot'_p\{'\square' \ seq \ '\bot'\}^* \ \{'\rightarrow (\square)'\}^? \ ']' \ \}^* \tag{3}$$
$$'[\bot_p\square' \ seq \ '\bot]'$$
$$\{ \ '[\bot'_p\{'\square' \ seq \ '\bot'\}^* \ \{'\rightarrow (\square)'\}^? \ ']' \ \}^*$$

$$par ::= \ '[\square' \ seq \ ']' \ \{'[\square' \ seq \ ']'\}^* \tag{4}$$
$$'[\square' \ seq \ ']' \qquad \diamond$$

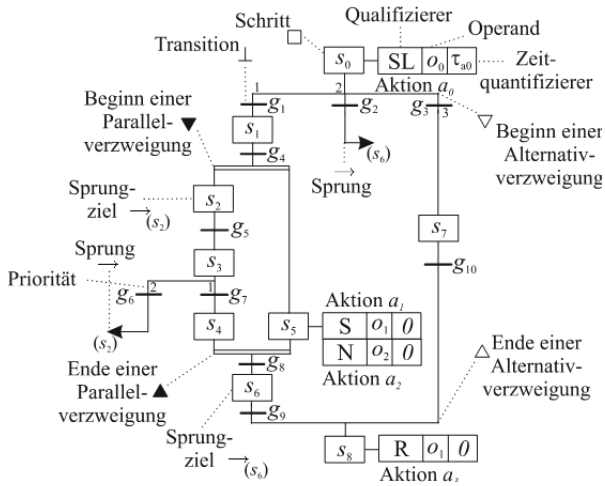

Abb. 2: Beispiel eines *AS*-Programms.

Regel (1) definiert das *AS*-Programm (ψ_{SFC}), Regel (2) Sequenzen (*seq*), Regel (3) Alternativverzweigungen (*alt*) und Regel (4) Parallelverzweigungen (*par*). Sprünge innerhalb des *AS*-Programms werden zwar nicht von der Norm unterstützt, aber von den meisten industriellen Implementierungen genutzt, sodass sie hier abgebildet werden. Eine Übersetzung in diese Syntax erzeugt für das in Abb. 2 gezeigte AS-Programm die folgende Zeichenkette:

$$\square_{(s_0)}\triangledown[\perp_{1(g_1)}\square_{(s_1)}\perp_{(g_4)}\blacktriangledown[\square_{(s_2)}\perp_{(g_5)}\square_{(s_3)}\triangledown[\perp_{2(g_6)} \rightarrow (\square_{(s_2)})][\perp_{1(g_7)}] \triangle \square_{(s_4)}][\square_{(s_5)}]\blacktriangle\perp_{(g_8)}\square_{(s_6)}\perp_{(g_9)}][\perp_{2(g_2)} \rightarrow (\square_{(s_6)})]_2[\perp_{3(g_3)}\square_{(s_7)} \perp_{(g_{10})}]_3 \triangle \square_{(s_8)}.$$

Die formale Semantik des *AS*-Formalismus wird durch die algorithmische Abbildung von *AS*-Programmen in das *Compositional Interchange Format* (*CIF*)[1] [7], ein automatenbasiertes Austauschformat für diskret-kontinuierliche Modelle, definiert [6]. Diese Abbildung erzeugt nicht nur ein genaues Modell des *AS*-Programms, sondern auch ein Verhaltensmodell der unterlagerten *speicherprogrammierbaren Steuerung* (*SPS*), einschließlich der zyklischen *SPS*-Arbeitsweise. Die algorithmische Abbildung von *AS*-Programmen in das *CIF*, sowie ein Direktexport in den *UPPAAL*-Formalismus, wurden im *DC/FT*-Werkzeug [1,2] implementiert.

4 Modellbasierte Validierung von *Ablaufsprache*-Programmen

Die Abbildung von *AS*-Programmen in das *CIF*-Format ermöglicht eine Kopplung des Steuerungsmodells mit Anlagenmodellen im *CIF* und die Weitertransformation in eine Vielzahl von modellbasierten Simulations-, Optimierungs- und Analysewerkzeugen (z.B. *gPROMS*, *Modelica*, *Matlab/Simulink*, *EcosimPro*, *UPPAAL*, *SpaceEx, MUSCOD II*, etc.). Diese Transformationen wurden im Rahmen des EU-Forschungsprojekts *MULTIFORM*[2] entwickelt und ermöglichen es dem Steuerungsingenieur, entworfene Steuerungen an der Anlage zu simulieren (z.B. in *gPROMS*, *Modelica*, *EcosimPro* oder *Matlab/Simulink*), Steuerungen an abstrakten Anlagenmodellen zu verifizieren (z.B. in *UPPAAL* oder *SpaceEx*) und Steuerungsparameter zu optimieren (z.B. in *gPROMS* oder *MUSCOD II*).

Danksagung: Die in diesem Beitrag vorgestellten Forschungsarbeiten wurden im Rahmen des europäischen Forschungsprojekts MULTIFORM, Vertragsnummer: INFSO-ICT-224249, im 7. Rahmenprogramm gefördert. Diese Unterstützung wird dankend anerkannt.

[1] S. Fischer, S. Engell: Werkzeugunterstützung für den Entwurf von Ablaufsteuerungen auf Basis informeller Spezifikationen. In *at- Automatisierungstechnik* 59, S. 50-61, 2011.

[2] S. Fischer, M. Hüfner, C. Sonntag, S. Engell: Systematic Generation of Logic Controllers in a Model-based Multi-formalism Design Environment. In *Proc. 18th IFAC World Congress*, S. 12490-12495, 2011.

[3] A. Hellgren, M. Fabian, B. Lennartson: On the Execution of SFCs. In *Control Engineering Practice* 13, S. 1283–1293, 2005.

[4] N. Bauer, R. Huuck, B. Lukoschus, S. Engell: A Unifying Semantics for Sequential Function Charts. In *Integr. of Software Spec. Techn. for Applications in Engin. 3147*, S. 400–418, 2004.

[5] O. Stursberg, S. Lohmann: Analysis of Logic Controllers by Transformation of SFC into Timed Automata. In *Proc. 44th Conf. on Decision and Control*, S. 7720–7725, 2005.

[6] C. Sonntag, S. Fischer: Translating Sequential Function Charts to the Compositional Interchange Format for Hybrid Systems. In *Proc. 49th Conf. on Decision and Control*, S. 4250-4256, 2010.

[7] D. A. van Beek, M.A. Reniers, R.R.H. Schiffelers, J.E. Rooda: Foundations of a Compositional Interchange Format for Hybrid Systems. In *Lecture Notes in Computer Science* 4416, Springer, S. 587-600, 2007.

[1] http://se.wtb.tue.nl/sewiki/cif

[2] http://www.ict-multiform.eu