http://dx.doi.org/10.25673/116044

Ansätze für die Modellierung von Streckeninformationen zur Verifikation im Steuerungsentwurf

Aron Schnakenbeck [1], Robin Mroß [2], Marcus Völker [2], Stefan Kowalewski [2] und Alexander Fay [3]

Abstract: Während der Entwicklung von automatisierten Anlagen ist die Spezifikation der Steuerungslogik ein wichtiger Schritt. Um diesen Prozess mit formalen Methoden zu unterstützen, kann die Steuerungslogik zunächst mithilfe eines formalen Modells spezifiziert werden, was eine formale Verifikation ermöglicht. In einem Vorgängerbeitrag der Autoren wurde eine solche Verifikation vorgestellt, die es ermöglicht, GRAFCET-Spezifikationen zu analysieren. In diesem Beitrag wird dieses Konzept um die Modellierung von Streckeninformationen erweitert. Die Modellierung von Streckeninformationen ermöglicht eine genauere Verifikation und in Teilen eine Reduktion des Zustandsraums. Anhand eines durchgängigen Beispiels werden zwei Ansätze für die Modellierung von Streckeninformationen vorgestellt und verglichen: die Modellierung mittels Invarianten und die Modellierung mittels GRAFCET. Bei der Modellierung mittels GRAFCET wird gezeigt, wie neben dem Verhalten der Steuerstrecke auch zusätzliche Eigenschaften des kombinierten Verhaltens modelliert werden können, wie beispielsweise die korrekte Abarbeitung des SPS-Zyklus. Um diese zusätzlichen Eigenschaften anwenderfreundlich modellieren zu können, wird eine Erweiterung von GRAFCET vorgestellt und gezeigt, wie dieses Konzept zur Modellierung von Streckeninformationen in die bereits vorgestellte Werkzeugkette zur Verifikation von GRAFCET-Spezifikationen integriert werden kann.

Keywords: GRAFCET, Verifikation, formale Methoden, Streckenmodellierung

1 Einleitung

Während der Entwicklung von automatisierten Anlagen ist die Spezifikation der Steuerungslogik ein wichtiger Schritt. Allerdings ist die in der Praxis verbreitete, manuelle Implementierung dieser Steuerungslogik aus informellen Anforderungen heraus fehleranfällig [SF14]. Formale Methoden können diesen Prozess unterstützen, indem zunächst die Steuerungslogik mithilfe eines formalen Modells spezifiziert wird. Dieses Vorgehen hat die Vorteile, dass die formale Spezifikation als Dokumentation für den Code genutzt werden kann und eine automatisierte Codegenerierung möglich wird [Ju19]. Zudem kann eine formale Verifikation der Steuerungslogik früh in den Entwicklungsprozess eingebunden

1 Helmut-Schmidt-Universität, Institut für Automatisierungstechnik, Holstenhofweg 85, 22043 Hamburg, aron.schnakenbeck@hsu-hh.de, https://orcid.org/0000-0002-3507-2851

2 RWTH Aachen University, Lehrstuhl Informatik 11, Ahornstraße 55, 52074 Aachen, mross@embedded.rwth-aachen.de, https://orcid.org/0000-0002-1007-5597; voelker@embedded.rwth-aachen.de, https://orcid.org/0000-0001-7348-0146; kowalewski@embedded.rwth-aachen.de, https://orcid.org/0000-0002-7184-4804

3 Ruhr-Universität Bochum, Universitätsstraße 150, 44801 Bochum, alexander.fay@rub.de, https://orcid.org/0000-0002-1922-654X

werden. In der Vergangenheit haben die Autoren dieses Beitrags in [Sc23] eine Werkzeugkette vorgestellt, die einen solchen formalen Entwicklungsprozess von Steuerungscode abbildet und eine formale Verifikation enthält. Allerdings berücksichtigt der vorgestellte Ansatz bei der Verifikation nur das Steuerungsverhalten. Das Verhalten der Steuerstrecke wurde nicht modelliert, sondern es wurde angenommen, dass sich die Eingangssignale der Steuerung nichtdeterministisch verhalten. Während der Evaluation dieses Modellierungsansatzes stellte sich heraus, dass gewisse Eigenschaften ohne weitere Informationen über das Streckenverhalten nicht definitiv nachgewiesen werden können. Dazu gehört beispielsweise, dass das einschränkende Verhalten der Steuerstrecke zu Verklemmungen führen kann, also dass es keine Transition mehr gibt, die noch schalten kann. Ein weiterer Vorteil das Streckenverhalten zu modellieren kann sein, dass es den Zustandsraum während der Verifikation einschränkt und so einer Zustandsraumexplosion entgegenwirken kann. Daher wird in diesem Beitrag die Modellierung von Streckenverhalten für die formale Verifikation diskutiert. Zudem wird gezeigt, wie die in [Sc23] vorgestellte Werkzeugkette erweitert werden kann, um die Modellierung des Streckenverhaltens während der Verifikation zu berücksichtigen.

Da die Akzeptanz von formaler Verifikation in der Praxis immer noch gering ist [Vo14], wird in diesem Beitrag, sowie in dem von den Autoren vorgestellten Verifikationsansatz [Sc23], GRAFCET als Beschreibungsmittel verwendet, das in der automatisierungstechnischen Praxis anerkannt ist [SF14]. GRAFCET ist eine in [DIN EN 60848] genormte, graphische Spezifikationssprache, die neben sequenziellem, alternativem und nebenläufigem Verhalten auch die Modellierung von hierarchischen Steuerungsstrukturen (mittels sogenannter *Teilgrafcets*) ermöglicht.

In ihrer Studie zu existierenden Modellierungsansätzen von Streckenverhalten in der Literatur zeigen Galvão et al. [Ga19], dass der Grad der Granularität entscheidend ist, mit dem das Streckenverhalten modelliert wird. Zum einen schreiben die Autoren, dass eine detailliertere Modellierung einen erhöhten Ressourcenbedarf bei der Verifikation erfordert und, dass der Modellierungsgrad beeinflusst, welche Eigenschaften verifiziert werden können. Daher wird in dieser Arbeit anhand eines durchgängigen Beispiels gezeigt, wie die Streckenmodellierung über zwei Ansätze verschiedener Granularität erfolgen kann: die Modellierung über boolesche Invarianten und die Modellierung mittels GRAFCET. Der Aufwand zur Erstellung eines vollständigen formalen Modells der Steuerstrecke ist allerdings hoch, und es ist nicht davon auszugehen, dass ein solches in der Praxis vorliegt. Daher ist das Ziel dieses Ansatzes die Modellierung von Teilinformationen, die für die Verifikation von besonderer Bedeutung sind, und in einer Form, die für den Anwender möglichst eingängig ist, was die Verwendung von beispielsweise temporalen Logiken ausschließt.

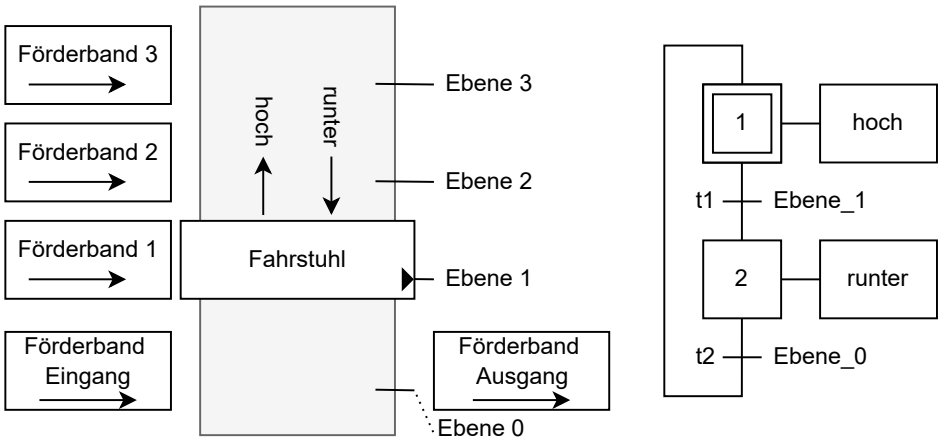

Abb. 1: Lastenfahrstuhl als Steuerstrecke und die zugehörige Steuerung als GRAFCET-Instanz.

2 Streckenmodellierung

Als Beispiel wurde ein Lastenfahrstuhl gewählt (siehe Abb. 1). Dieser hat auf vier verschiedenen Ebenen Förderbänder als Eingänge und auf der untersten Ebene ein Förderband als Ausgang. Der Fahrstuhl verfügt über zwei Aktoren, die den Fahrstuhl herauf- und herunterbewegen, und mehrere Sensoren, die die Anwesenheit des Fahrstuhls in einer bestimmten Höhe registrieren. Die vereinfacht dargestellte Steuerung sorgt dafür, dass der Fahrstuhl nach oben fährt, bis die erste Ebene erreicht ist, und im Anschluss nach unten fährt, bis die unterste Ebene erreicht ist.

2.1 Modellierung von Streckeneigenschaften mit Invarianten

Der Zusammenhang zwischen den Sensorvariablen, der sich aus der Geometrie der Strecke ergibt, kann über Invarianten in Form von einfachen booleschen Bedingungen formuliert werden. Im Beispiel des Fahrstuhls (siehe Abb. 1) kann zu jedem Zeitpunkt nur maximal einer der Sensoren den Fahrstuhl registrieren. So ist die entsprechende Variable genau dann 1, wenn der Fahrstuhl registriert wird, anderenfalls 0:

$$(Ebene\_0 + Ebene\_1 + Ebene\_2 + Ebene\_3 \leq 1) = wahr \tag{1}$$

Eine wichtige Anforderung für den Steuerungsentwurf ist, dass die Steuerung immer eine nächste stabile Situation erreichen kann. Für die Steuerung in Abb. 1 ist das gegeben, wenn die beiden Transitionsbedingungen nicht gleichzeitig *wahr* sein können, also `Ebene_0` und `Ebene_1` nicht gleichzeitig *wahr* sein können. Die Verifizierung dieser Anforderung wird unter der Berücksichtigung der Invariante möglich. Vergleichbares gilt bei der Verifikation,

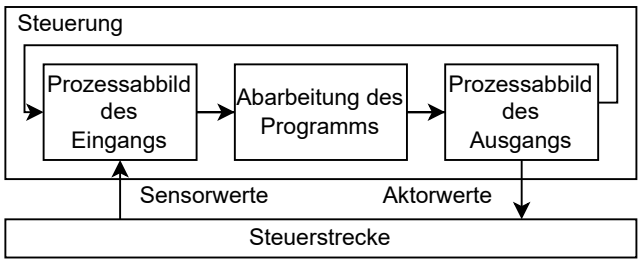

Abb. 2: Kombiniertes Verhalten aus Steuerung und Steuerstrecke. Darstellung nach [Lu20].

dass Alternativverzweigungen nur exklusiv schalten können, wie von der Norm [DIN EN 60848] vorgegeben.

In vorherigen Arbeiten der Autoren [Sc23] wurden Transformationsregeln vorgestellt, die Grafcets in Modelle formuliert in Guarded Action Language (GAL) [Th16] überführen, mit dem Ziel der Modellverifikation, mittels des Model Checkers ITS-Tools [Th15]. Die Invarianten, die die Streckeninformationen abbilden, müssen zum Zeitpunkt der Erzeugung des GAL-Modells berücksichtigt werden. In der bisherigen Transformation nach GAL wird eine Komponente generiert, die jeder booleschen Eingangsvariable einen zufälligen Wert zuweist. Diese Zuweisung ist - ohne Wissen über die Strecke - nichtdeterministisch und erzeugt im resultierenden Transitionssystem alle möglichen Belegungen dieser Variablen als Folgezustände. Invarianten wie in Gleichung 1 schränken die möglichen Folgezustände ein: Jeder Folgezustand, der diese Invarianten nicht erfüllt, darf nicht erzeugt werden. Um sicherzustellen, dass diese nicht erzeugt werden, wird nach der Zuweisung der neuen Werte überprüft, ob die Invariante verletzt wird. Wenn dies der Fall ist, so werden die involvierten Eingangsvariablen so gesetzt, dass die Invariante erfüllt wird. Wenn zum Beispiel die Eingänge *Ebene_0*, *Ebene_1*, *Ebene_2* und *Ebene_3* den Wert *wahr* erhalten haben, dann genügt es all diese Variablen auf *falsch* zu setzen. Für den Fall, dass eine Invariante unerfüllbar ist, oder dass mehrere Invarianten im Widerspruch zueinander stehen, kann in GAL Gebrauch von dem Prädikat *abort* gemacht werden, welches keinen Folgezustand generiert. Das resultierende Modell ist dann allerdings der Art, dass eine in der Regel nicht wünschenswerte Verklemmung vorliegt. Dies kann anschließend mittels Model Checking überprüft werden.

2.2 Modellierung von Streckeneigenschaften mit GRAFCET

Eine weitere Anforderung an das System in Abbildung 1 kann sein, dass es in der Steuerung zu keiner Verklemmung kommen darf, also immer eine nächste Situation erreichbar ist. Dies ist nicht gegeben, wenn keine Transition mehr schalten kann. Angenommen, während der Spezifikation in Abbildung 1 entsteht ein Fehler und die Aktion an Schritt 2 setzt ebenfalls die Variable *hoch* auf *wahr*, dann ist zu beobachten, dass *Ebene_0* niemals

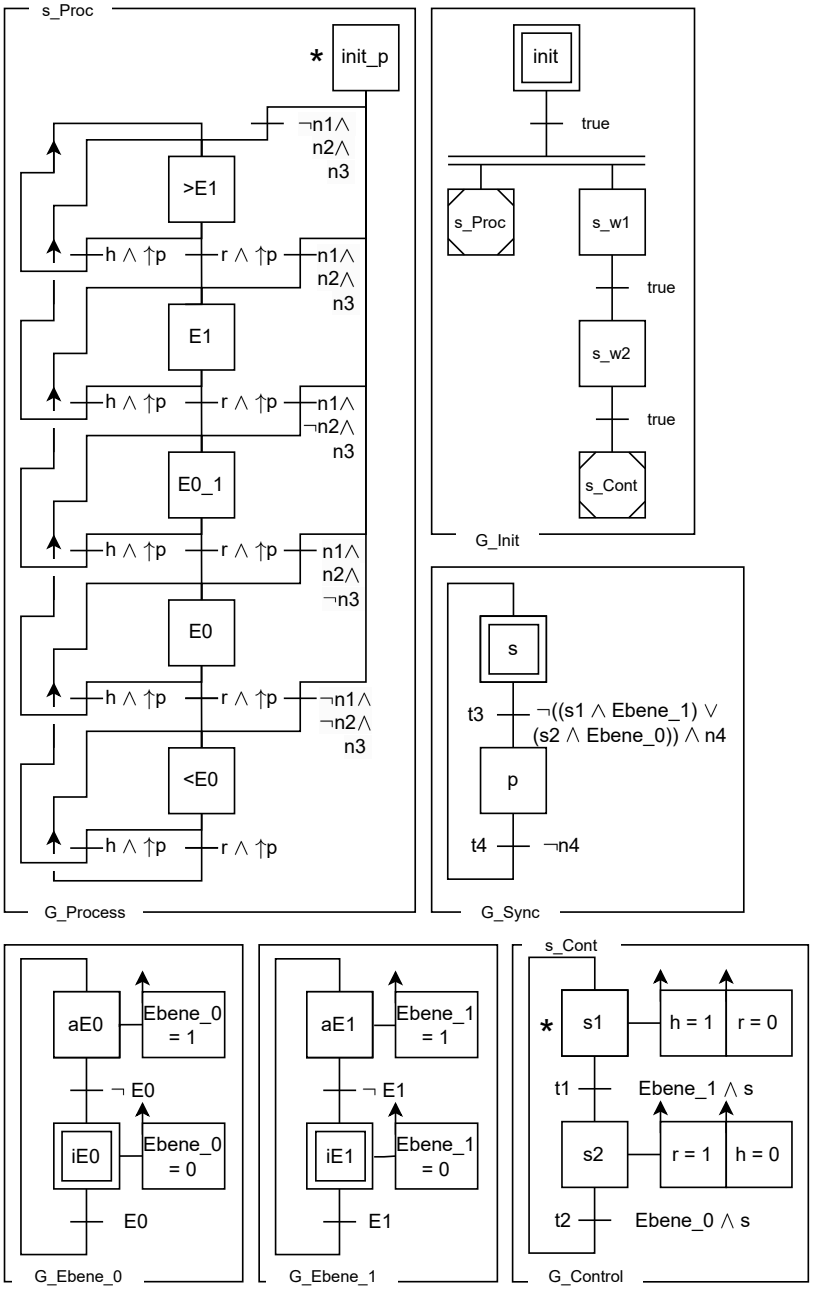

Abb. 3: In GRAFCET modelliertes kombiniertes Verhalten der in Abbildung 1 dargestellten Steuerung und Strecke mit *r* für runter und *h* für hoch.

wahr wird, da der Fahrstuhl niemals so angesteuert wird, dass er nach unten fährt. Die Invariante in Gleichung 1 ist nicht geeignet, diese Anforderung zu verifizieren, da sie keine Auskunft über den Zusammenhang zwischen Sensorik und Aktorik gibt. Um diese benötigten Informationen zu modellieren, soll in dieser Arbeit GRAFCET sowohl für die Modellierung der Steuerung als auch für die Modellierung der Steuerstrecke genutzt werden. Dies hat den Vorteil, dass der Anwender durchgängig ein Beschreibungsmittel verwenden kann. Um das kombinierte Verhalten aus Steuerung und Strecke innerhalb eines Grafcets[4] korrekt modellieren zu können, muss zusätzlich zum (I) Steuerungscode auch das Verhalten der (II) Strecke und das (III) Zusammenspiel aus Steuerung und Strecke mit GRAFCET modelliert werden. In Abbildung 2 ist das Zusammenspiel der Komponenten dargestellt.

Für das Beispiel des Lastenfahrstuhls ist die Modellierung des (I) Steuerungscodes bereits in Abbildung 1 erfolgt und wird im Folgenden noch geringfügig angepasst. Die Modellierung der Strecke (II) ist in Abbildung 3 in dem Teilgrafcet *G_Process* dargestellt. Dabei bildet das Modell nicht das vollständige Verhalten des Lastenfahrstuhls ab, sondern nur die für die Verifikation nützlichen Informationen. Der Fahrstuhl wird mit fünf Schritten modelliert, die anzeigen, ob der Fahrstuhl von einem Sensor detektiert wird (*E1*, *E0*), oder ob er sich unter, zwischen, oder über den Sensoren befindet (*<E0*, *E0_1*, *>E1*). Je nachdem, ob die Steuerung die Befehle *hoch (h)* oder *runter (r)* sendet, wird ein entsprechend anderer Schritt des Strecken-Grafcets aktiviert. GRAFCET ist dazu entworfen, deterministisches Verhalten zu modellieren, was für den Entwurf von Steuerungen auch angemessen ist, da nichtdeterministisches Verhalten in Steuerungen üblicherweise unerwünscht ist. Steuerstrecken können sich aber durchaus nichtdeterministisch verhalten, beispielsweise wenn sich eine Strecke in einer nicht definierten Ausgangssituation befindet. Um dies abzubilden, wird ein einzelner Anfangsschritt *init_p* definiert, der einen der Schritte der Strecke über eine Transition aktiviert. Dies wird zum einen erreicht, da sich die Transitionsbedingungen dieser Transitionen gegenseitig ausschließen und so immer nur einer der Schritte des Strecken-Grafcets aktiviert wird. Dass während der Verifikation im Zustandsraum alle möglichen initialen Situationen der Strecke abgedeckt sind, wird dadurch sichergestellt, dass die Transitionsbedingungen, die auf den Initialschritt folgen, mit als nichtdeterministisch modellierten Eingangsvariablen belegt sind.

Für die Modellierung des Zusammenspiels (III) zwischen Strecke und Steuerung wird Teilgrafcet *G_Sync* eingefügt. Dieser hat zwei Schritte *s* und *p*, die dafür sorgen, dass sich entweder die Strecke oder die Steuerung entwickeln kann. Auf Seiten der Steuerung garantiert dies, dass keine Zwischenwerte während der Abarbeitung des Steuerungscodes an die Strecke gelangen (wie in Abbildung 2 dargestellt). Die Transitionsbedingungen in *G_Control* werden dann um das Konjunkt der Schrittvariable *s* erweitert. Auf Seiten der Strecke kann umgesetzt werden, dass immer nur ein Zustandsübergang zurzeit stattfinden kann, auf den die Steuerung dann wieder reagiert, indem die Transitionsbedingungen in *G_Process* um das Konjunkt $\uparrow p$[5] erweitert werden. Dieses Verhalten modelliert dann

4 *GRAFCET* bezeichnet das Beschreibungsmittel und *Grafcet* eine Instanz des Beschreibungsmittels.

5 Der Operator $\uparrow x$ wird genau dann *wahr*, wenn eine steigende Flanke im Signal *x* auftritt.

eine realitätsnahe Synchronisation zwischen Steuerung und Strecke. Ein Wechsel von der Abarbeitung der Steuerung zur Abarbeitung der Strecke und umgekehrt erfolgt immer dann, wenn *G_Control* oder *G_Process* eine stabile Situation erreicht haben. Dies wird über die Transitionsbedingungen in *G_Sync* sichergestellt.

Zudem sind noch syntaktische Regeln von GRAFCET zu berücksichtigen. Werden Steuerung und Strecke in einer GRAFCET-Instanz abgebildet, so können Eingangs- und Ausgangsvariablen in GRAFCET, die üblicherweise zu Sensorik und Aktorik korrespondieren, nicht wie gewohnt verwendet werden, da die Eingangsvariablen der Steuerung die Ausgangsvariablen der Strecke sind (siehe Abbildung 2). Die Variablen, mit denen Steuerung und Strecke kommunizieren, werden daher in dem hier vorgestellten Ansatz in interne Variablen umgewandelt. Dies hat die Konsequenz, dass kontinuierlich wirkende Aktionen nicht mehr verwendet werden können, da diese syntaktisch nur Ausgangsvariablen schreiben können. Jede Variable, die mit kontinuierlich wirkenden Aktionen geschrieben wird, wird daher durch ein Konstrukt substituiert, das in Abbildung 3 für die Variablen *Ebene_0* und *Ebene_1* mit *G_Ebene_0* und *G_Ebene_0* dargestellt ist. Dies hat nun die Folge, dass es in der internen Abarbeitung von GRAFCET zwei sogenannte *transiente* Abläufe dauert, bis die Variablen (hier *Ebene_0* und *Ebene_1*) korrekt beschrieben werden. Daher wird zur korrekten Initialisierung das Konstrukt *G_Init* eingeführt, welches *G_Prozess* und *G_Control* aktiviert, wenn die Schritte *s_Proc* und *s_Cont* aktiv sind.

Das dargestellte Beispiel bildet eine normkonforme GRAFCET-Instanz, die das kombinierte Verhalten von Steuerung und Strecke korrekt wiedergibt, wobei die Modellierung von physikalischer Zeit allerdings nicht betrachtet wird.

Es ist festzustellen, dass die Modellierung der in Abbildung 3 dargestellten Konstrukte einen erheblichen Modellierungsaufwand bedeuten würde, welcher aber mittels einer Modelltransformation automatisiert werden kann. Dazu schlagen wir in diesem Beitrag eine Erweiterung von GRAFCET vor, die wir in Anlehnung an Frey et al. [FL98] als *Prozesstechnisch Interpretierter GRAFCET* bezeichnen und die eine auf die Strecke ausgerichtete Syntax und Semantik enthält, wobei die Semantik im Wesentlichen über die beispielhaft vorgestellte Modelltransformation definiert ist. Dieser *Prozesstechnisch Interpretierte GRAFCET* kann dann über eine Modelltransformation in einen regulären GRAFCET nach [DIN EN 60848] transformiert werden, wie in Abbildung 3 für das Beispiel gezeigt, sodass sich der Nutzer nicht mit der Modellierung des SPS-Steuerzyklus beschäftigen muss. Formal handelt es sich um eine Erweiterung des GRAFCET-Metamodells wie in [Sc23] vorgestellt, dem eine Klasse für den *Prozesstechnisch Interpretierten GRAFCET* hinzugefügt wird, die einzig anzeigt, dass in ihr eine Strecke modelliert wird, sodass eine Transformation möglich wird. Die entstehenden Grafcets können im Anschluss mit der bestehenden, in [Sc23] vorgestellten, Werkzeugkette verifiziert werden.

Zudem ist in Bezug auf den Modellierungsaufwand zu beachten, dass es sich bei den für die Verifikation interessanten Streckeninformationen um recht einfache Zusammenhänge zwischen Sensorvariablen handeln kann, die sich aus der Streckengeometrie ergeben. Dies

ist beispielsweise bei Endlagensensoren von Kolben, bei der Detektion mehrerer Füllstände eines Tanks, oder bei dem gezeigten Fahrstuhl der Fall. All diese Streckeninformationen sind in GRAFCET ähnlich modellierbar, sodass eine Art Bibliothek definiert werden kann, aus der der Anwender das entsprechende GRAFCET-Konstrukt auswählt und nur noch die Variablenbenennung vornehmen muss. Die entsprechende Transformation des *Prozesstechnisch Interpretierten GRAFCET* erfolgt dann automatisiert in normkonformen GRAFCET.

3 Diskussion

Eine fehlende Modellierung von einschränkenden Streckeninformationen führt zur größtmöglichen Überapproximation des Streckenverhaltens, was für falsche Alarme in der Verifikation sorgen kann. Mit beiden vorgestellten Ansätzen wird diese Überapproximation verringert. Für Invarianten ist die Überapproximation aber höher, als bei der Modellierung mittels GRAFCET, da die Invarianten keine Zustände abbilden können. Es konnte beispielhaft gezeigt werden, dass die Zuhilfenahme von Invarianten dazu führt, genauere Verifikationsergebnisse zu erzielen, wenn transiente Abläufe oder die Exklusivität von Alternativverzweigungen analysiert werden. Mit der Streckenmodellierung als Grafcet konnten diese beiden Anforderungen ebenfalls genauer verifiziert werden, und darüber hinaus, ob es im kombinierten Verhalten aus Steuerung und Strecke zu Verklemmungen kommt. Weiter kann die Modellierung mittels Grafcet dazu genutzt werden, die Strecke betreffende Anforderungen zu verifizieren, wie beispielsweise die Erreichbarkeit von Zuständen in der Strecke. Für den Lastenfahrstuhl wäre eine solche Anforderung, dass der Fahrstuhl die obere/untere Endlage nicht über-/unterschreiten darf. In Bezug auf den Modellierungsaufwand sind die vorgestellten Invarianten durch den Anwender vergleichsweise einfach in boolescher Logik zu formulieren. Die Beschreibung der Strecke mit GRAFCET ist für einfache Zusammenhänge unter Nutzung der Modelltransformation kein großer zusätzlicher Modellierungsaufwand, da für ähnliche Steuerungsgeometrien die Grafcets auch wiederverwendet werden können. Sollen allerdings komplexere Streckenmodelle umgesetzt werden, ist auch ein höherer Modellierungsaufwand nötig.

Die Modellierung der Strecke als normkonformen GRAFCET hat hingegen den Vorteil, dass sie unabhängig von der Implementierung der Verifikationsmethode ist, sodass jedes bestehende GRAFCET-Verifikationswerkzeug auch auf einen Grafcet angewendet werden kann, welcher Streckeninformationen enthält. Für die Invarianten hingegen muss beim Model Checking während der Generierung des Zustandsraums sichergestellt werden, dass sie auch gelten, was sich nur werkzeugspezifisch umsetzen lässt.

Für das Ziel der Zustandsraumverkleinerung ist die Formulierung von Invarianten vorteilhaft, da sie die Kombinatorik der Eingangsvariablen tendenziell einschränken. Für die Modellierung von Streckeninformation in GRAFCET ist dies allerdings nicht der Fall. Für das gezeigte Beispiel vergrößert sich der Zustandsraum um ein Vielfaches.

4 Zusammenfassung und Ausblick

Es wurden zwei Ansätze zur Modellierung von Streckeninformation vorgestellt: Invarianten und die Modellierung mittels GRAFCET. Für Invarianten wurde dargelegt, wie diese beispielhaft formuliert und dann bei der Erstellung des Zustandsraums in GAL für das Model Checking berücksichtigt werden können. Für die Modellierung mittels GRAFCET wurde dargelegt, wie neben den Streckeninformationen noch weitere Konstrukte modelliert werden müssen, sodass das kombinierte Verhalten aus Steuerung und Strecke sichergestellt werden kann. Dies ermöglicht die Modellierung des SPS-Zyklus und die Einhaltung oder auch Verletzung von Echtzeitkriterien. Um dies mittels einer Modelltransformation zu unterstützen, wurde der *Prozesstechnisch Interpretierte GRAFCET* eingeführt, in dem das Streckenverhalten modelliert wird. In der Diskussion wurde gezeigt, dass die Modellierung mittels GRAFCET zu bevorzugen ist, wenn die höchste Genauigkeit der Verifikationsergebnisse erzielt werden soll. Invarianten eignen sich, wenn der Modellierungsaufwand gering gehalten werden soll, oder wenn eine Reduzierung des Zustandsraums erzielt werden soll.

Für zukünftige Arbeiten ist es sinnvoll, die Verifikation des Zeitverhaltens von GRAFCET zu untersuchen, was in [Sc23] nicht berücksichtigt wurde. Dies ermöglicht es während der Verifikation auch das Zeitverhalten der Strecke zu berücksichtigen. Auch kann es sinnvoll sein, die Streckenmodellierung basierend auf Zuständen in dem Formalismus durchzuführen, der für das Model Checking genutzt wird (in diesem Fall GAL), um eine Begrenzung der Zustandsraumgröße zu erzielen. Auch kann es sinnvoll sein, die Streckenmodelle aus dem vorgestellten *Prozesstechnisch Interpretierten GRAFCET* direkt in GAL zu transformieren, um eine bessere Kontrolle über die Zustandsgröße zu erhalten und diese möglicherweise stärker zu begrenzen als in dem vorgestellten Ansatz.

Literatur

[DIN EN 60848] DIN Deutsches Institut für Normung e. V.: GRAFCET, Spezifikationssprache für Funktionspläne der Ablaufsteuerung, DIN EN 60848, 2014-12.

[FL98] Frey, G.; Litz, L.: Verification and validation of control algorithms by coupling of interpreted Petri nets. In: SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics. Bd. 1, 7–12 vol.1, 1998.

[Ga19] Galvão, J.; Oliveira, C.; Lopes, H.; Tiainen, L.: Formal Verification: Focused on the Verification Using a Plant Model. In (Machado, J.; Soares, F.; Veiga, G., Hrsg.): Innovation, Engineering and Entrepreneurship. Springer International Publishing, Cham, S. 124–131, 2019.

[Ju19] Julius, R.; Fink, V.; Uelzen, S.; Fay, A.: Konzept zur bidirektionalen Transformation zwischen GRAFCET-Spezifikationen und IEC 61131-3 Steuerungscode. at - Automatisierungstechnik 67/3, S. 208–217, 2019.

[Lu20] Lunze, J.: Automatisierungstechnik - Methoden für die Überwachung und Steuerung kontinuierlicher und ereignisdiskreter Systeme. De Gruyter, Berlin, Boston, 2020, ISBN: 9783110465624.

[Sc23] Schnakenbeck, A.; Mroß, R.; Völker, M.; Kowalewski, S.; Fay, A.: Transformation von GRAFCET in GAL auf Basis eines ausführlichen Metamodells zur Verifikation von Entwurfsfehlern. at - Automatisierungstechnik 71/1, S. 56–68, 2023.

[SF14] Schumacher, F.; Fay, A.: Petrinetzmodell für die Formalisierung von GRAFCET-Spezifikationen. at - Automatisierungstechnik 62/6, S. 385–393, 2014.

[Th15] Thierry-Mieg, Y.: Symbolic Model-Checking Using ITS-Tools. In (Baier, C.; Tinelli, C., Hrsg.): Tools and Algorithms for the Construction and Analysis of Systems. Springer Berlin Heidelberg, Berlin, Heidelberg, S. 231–237, 2015.

[Th16] Thierry Mieg, Y.: From Symbolic Verification To Domain Specific Languages, Diss., Sorbonne Université , UPMC ; Laboratoire d'informatique de Paris 6 [LIP6], 2016.

[Vo14] Vogel-Heuser, B.; Diedrich, C.; Fay, A.; Jeschke, S.; Kowalewski, S.; Wollschlaeger, M.; Göhner, P.: Challenges for Software Engineering in Automation. Journal of Software Engineering and Applications 7/, S. 440–451, 2014.