Partner datenverarbeitender Services

Christoph Wagner

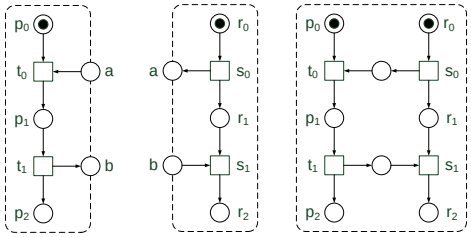
Institut für Informatik, Humboldt Universität zu Berlin,
Unter den Linden 6, 10099 Berlin, Germany
cwagner@informatik.hu-berlin.de

Zusammenfassung Offene Netze sind eine petrinetzbasierte formale Beschreibung von Services. Zwei offene Netze sind füreinander Partner, wenn ihre Komposition aus jedem erreichbaren Zustand einen Endzustands erreichen kann. Dieser Beitrag erweitert das Konzept der offenen Netze auf High-Level Petrinetze, um die Verarbeitung von Daten in Services darzustellen. Es werden exemplarisch Partner offener High-Level Netze und die in ihnen vorkommenden Ausdrucksmittel untersucht.

1 Einleitung

Service Oriented Computing (SOC) beschreibt ein Paradigma zum Aufbau verteilter Systeme aus Services. Ein *Service* ist eine eigenständige funktionale Einheit, die durch asynchrone Nachrichtenkanäle mit anderen Services kommunizieren kann. *Offene Netze* [2,3] sind ein Mittel zur formalen Beschreibung von Services. Mit offenen Netzen lassen sich Kommunikation und Komposition von Services auf natürliche Weise im Petrinetzkalkül ausdrücken.

Ein offenes Netz ist ein Petrinetz mit speziell ausgezeichneten *Sende-* und *Empfangsplätzen*, welche Puffer für asynchrone Nachrichtenkanäle repräsentieren. Abb. 1 zeigt ein offenes Netz N_1 mit einem Empfangsplatz a und einem Sendeplatz b. Wir verzichten im Folgenden auf eine formale Definition und verweisen für technische Details auf [1,2]. Zwei offene Netze N_1 und N_2 heißen (syntaktisch) *kompatibel*, wenn jeder



(a) N_1 (b) N_2 (c) $N_1 \oplus N_2$

Abbildung 1: Kompatible offene Netze und ihre Komposition

Sendeplatz eines Netzes ein Empfangsplatz des jeweils anderen Netzes ist. Kompatible offene Netze werden *komponiert*, indem gleichnamige Empfangs- und Sendeplätze

verschmolzen werden. Die verschmolzenen Plätze werden dabei zu internen Plätzen der Komposition $N_1 \oplus N_2$. Die Komposition ist ein *geschlossenes Netz* ohne Sende- und Empfangsplätze.

Für ein einzelnes offenes Netz stellt sich die Frage nach seiner *Bedienbarkeit*: Gegeben ein offenes Netz N, gibt es ein kompatibles offenes Netz S, so dass die Komposition $N \oplus S$ aus jedem erreichbaren Zustand einen ausgezeichneten Endzustand erreichen kann? Eine Ursache von Nichtbedienbarkeit können in der Komposition erreichbare Deadlocks sein. Bedienbarkeit ist ein Grundproblem der Controllersynthese und wurde im Kontext von offenen Netzen z. B. in [2,3] untersucht. Der den Analyseverfahren zugrunde liegende Formalismus berücksichtigt die in den Nachrichten enthaltenen Datenwerte bisher nicht explizit. Offene Netze, die große oder gar unendlich große Domänen verwenden, können mit rechnergestützten Methoden nur schwer oder gar nicht analysiert werden.

Dieser Beitrag verfolgt den Ansatz, Datenwerte explizit in den Formalismus der offenen Netze mit einzubeziehen. Dafür erweitern wir das Konzept der offenen Netze auf High-Level Petrinetze. Datenwerte werden in einem High-Level Petrinetz durch farbige Marken dargestellt. Da die im Folgenden behandelten Probleme weitgehend unabhängig vom verwendeten High-Level Petrinetzformalismus sind, soll die Definition hier nur kurz skizziert werden:

Ein High-Level Petrinetz besteht aus einer Menge von Plätzen P, einer Menge von Transitionen T, einer Menge F von Kanten und einer Anfangsmarkierung m_0. Jedem Platz p ist eine eine Menge von Werten (seine *Domäne*) $dom(p)$ zugeordnet. Jeder Transition t ist ein *Guard* $g(t)$ und eine Menge $var(t)$ von *Schaltvariablen* zugeordnet, wobei jede Variable x ebenfalls eine Domäne $dom(x)$ hat. Die Kanten sind mit Termen über den Schaltvariablen beschriftet. Eine *Markierung* m ist eine Funktion, die jedem Platz p eine Multimenge von Elementen aus $dom(p)$ zuweist. Eine Funktion β, die jede Variable $x \in var(t)$ einer Transition t mit einem Element $e \in dom(x)$ belegt, heißt *Schaltmodus* von t. Beim Schalten einer Transition in Modus β werden die Kantenbeschriftungen mit β ausgewertet und entsprechend Marken von Vorplätzen konsumiert bzw. auf den Nachplätzen produziert. Voraussetzung zum Schalten ist, dass der Guard für β zu $true$ auswertet.

Der folgende Abschnitt erläutert zunächst den Begriff des *Partners* und zeigt exemplarisch einige Partner von offenen High-Level Petrinetzen. Es wird untersucht, wie sich die Verwendung von Variablen in einem offenen Netz auf die Struktur seiner Partner auswirkt. Es zeigt sich, dass sich selbst bei Verzicht auf Ausdrucksmittel wie Guards, Funktionssymbole und Konstanten nicht-triviale Abhängigkeiten ergeben, die ein Partner berücksichtigen muss.

2 Partner eines datenverarbeitenden Services

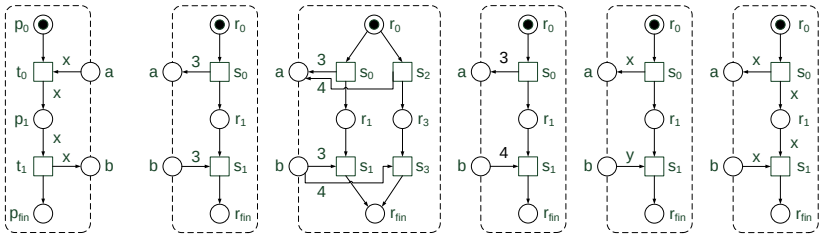
Wir betrachten im Folgenden die Partner von offenen High-Level Netzen. Zwei offene Netze mit jeweils ausgezeichneten Mengen von *Endmarkierungen* nennen wir *Partner*, wenn ihre Komposition *schwach terminiert* (vergleiche hierzu Def. 4 in [2]).

Definition 1 (Partner). *Ein (geschlossenes) Petrinetz* terminiert schwach*, wenn von jeder erreichbaren Markierung aus eine Endmarkierung des Netzes erreichbar ist. Seien*

N, S kompatible offene Netze. Wir nennen S einen Partner *von N, wenn die Komposition $N \oplus S$ schwach terminiert. Die Menge der Partner von N notieren wir mit $Partner(N)$.*

Die Endmarkierungen der Komposition ergeben sich kanonisch aus den Endmarkierungen der komponierten offenen Netze. Man beachte, dass $N \oplus S$ nicht schwach terminiert, wenn $N \oplus S$ einen Deadlock enthält. Das folgende Beispiel dient zunächst der Erläuterung des Partnerbegriffs. Die einzige Endmarkierung der offenen Netze bestehe jeweils aus einer Marke auf dem Platz p_{fin} bzw. r_{fin}. Die Domäne jedes Platzes sei entweder $\mathbb{N}$ oder $\{\bullet\}$, wobei $\bullet$ das Symbol für eine Marke ist, die keinen Wert trägt. Jede Variable habe die Domäne $\mathbb{N}$.

Beispiel 1 Das offene Netz N aus Abb. 2 setzt eine Nachrichtenweiterleitung um. Die von N auf a (durch Schalten von t_0) empfangene Nachricht wird unverändert (durch Schalten von t_1) auf b zurückgeschickt. Das offene Netz S_1 ist ein Partner von N. S_1



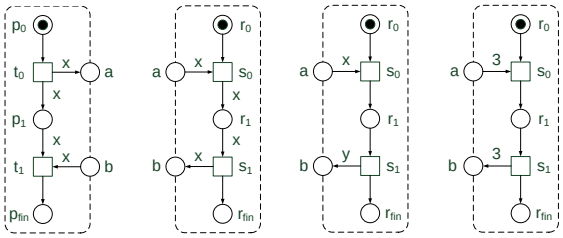
(a) N (b) S_1 (c) S_2 (d) S_3 (e) S_4 (f) S_5

Abbildung 2: Ein offenes Netz N und kompatible offene Netze

sendet zunächst durch Schalten von s_0 eine Nachricht auf a mit dem Wert 3. Da N auf b den selben Wert 3 sendet, kann auch s_1 schalten. Beide offene Netze erreichen somit ihre Endmarkierungen $[p_{fin}]$ und $[r_{fin}]$. Daher terminiert $N \oplus S_1$ schwach. Das offene Netz S_2 ist ebenfalls ein Partner von N. Es kann alternativ zum Wert 3 auch den Wert 4 senden und und diesen anschließend empfangen. Das offene Netz S_3 ist dagegen kein Partner von N. Es sendet den Wert 3 auf a, kann auf b aber nur Nachrichten mit Wert 4 empfangen. Da N auf b jedoch den Wert 3 sendet, kann s_1 nicht schalten und die Endmarkierung $[r_{fin}]$ von S_3 wird nicht erreicht. Das offene Netz S_4 ist ein Partner von N. Es wählt zunächst nichtdeterministisch eine natürliche Zahl als Belegung für x und sendet diese auf a. Anschließend empfängt es einen beliebigen Wert auf b (unabhängig davon, ob dieser gleich dem gesendeten ist) und wechselt in seine Endmarkierung $[r_{fin}]$. S_5 verhält sich auf die gleiche Weise, verlangt jedoch, dass der auf b empfangene Wert gleich dem auf a gesendeten Wert sein *muss*. S_5 ist in diesem Sinne präziser als S_4: Beide offene Netze sind gleichermaßen Partner von N, jedoch lässt die Struktur von S_5 einen genaueren Rückschluss auf die Funktionsweise von N zu. Aus S_4 kann man nicht erkennen, dass der von N auf b gesendete Wert stets gleich dem von N auf a empfangenen Wert ist.

Das nächste Beispiel zeigt ein für High-Level Netze spezifisches Phänomen auf.

Beispiel 2 Das offene Netz N' in Abb. 3 hat die gleiche Struktur wie S_2 aus Abb. 2. Es ähnelt dem offenen Netz N aus Abb. 2, jedoch sind Eingabe und Ausgabe vertauscht. Trotz dieses scheinbar geringen Unterschieds verfügt N' über zwei entscheidende Ausdrucksmittel, über die N nicht verfügt:



(a) N' (b) S'_1 (c) S'_2 (d) S'_3

Abbildung 3: Ein offenes Netz N' und kompatible offene Netze

1. t_0 ist eine *erzeugende* Transition, d. h. sie führt einen neuen Wert in das Netz ein. Die Variable x ist beim Schalten von t_0 nicht gebunden an den Wert einer Marke auf einem Eingangsplatz. Vielmehr wird x nichtdeterministisch an einen beliebigen Wert seiner Domäne $dom(x)$ gebunden. Im Unterschied zu N, wo t_0 einen auf a bereits vorhandenen Wert auf p_1 lediglich verschiebt, wird in N' auf den Plätzen p_1 und a ein Wert neu *erzeugt*.
2. t_1 führt einen *Test auf Gleichheit* aus. t_1 kann nur dann schalten, wenn auf p_1 und b Marken mit gleichen Werten liegen. Somit bestimmt der von der Umgebung auf b gesendete Wert direkt, ob N' seine Endmarkierung $[p_{fin}]$ erreichen kann oder nicht.

Über einen Partner von N' können wir folgern: Der vom Partner auf b gesendete Wert darf nicht unabhängig sein vom auf a empfangenen Wert. Vielmehr muss ein Partner Sorge dafür tragen, dass der von ihm auf b gesendete Wert gleich dem von ihm auf a empfangenen ist. Aus diesem Grund ist S'_1 ein Partner von N', S'_2 jedoch nicht. S'_3 berücksichtigt im Unterschied zu S'_2 die Gleichheit von empfangenem und gesendeten Wert, deckt aber nur einen einzigen Wert aus $dom(x)$ ab. Daher ist s_0 für die meisten auf a empfangbaren Werte nicht aktiviert und S'_3 infolgedessen kein Partner von N'.

Da $dom(x) = \mathbb{N}$ eine unendlich große Menge ist, muss jeder Partner ebenfalls über ein Ausdrucksmittel verfügen, das einen unendlich großen Wertebereich abdeckt. Diese Folgerung formulieren wir wie folgt:

Vermutung 1. Jeder Partner von N' enthält eine Kante, die mit einer Variablen beschriftet ist, deren Wertebereich unendlich groß ist.

Eine weitere entscheidende Beobachtung ist, dass ein Partner offenbar gewisse *Abhängigkeiten* zwischen gesendeten und empfangenen Werten einhalten muss. Augenscheinlicher Verursacher dieses Phänomens ist die Transition t_1. Diese ist eine *datenabhängige*

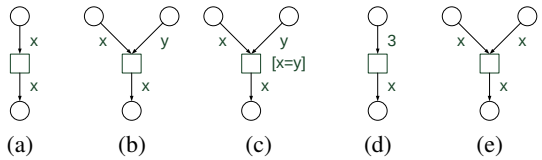
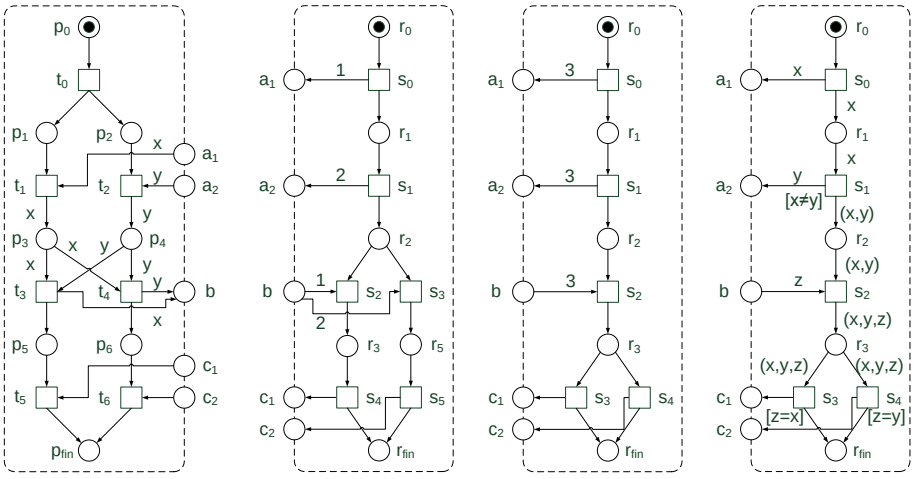


Abbildung 4: Datenunabhängige Transitionen: (a) - (b). Datenabhängige Transitionen: (c) - (e).

Transition. Wir nennen eine Transition datenabhängig, wenn die Entscheidung, ob sie schalten darf, von den Werten der Marken auf ihren Vorplätzen abhängt. Umgekehrt ist eine Transition *datenunabhängig*, wenn die Entscheidung nur von der Anzahl der Marken auf den Vorplätzen abhängt. Die Werte der produzierten Marken dürfen dagegen sehr wohl von den Werten der konsumierten Marken abhängen. Abb. 4 zeigt einige datenabhängige und -unabhängige Transitionen. Transitionen mit Guards sind im allgemeinen datenabhängig. Es ist offensichtlich, dass eine datenunabhängige Transition genau dan schalten kann, wenn sie im Skelett des Netzes schalten kann.

Das nächste Beispiel zeigt, dass jedoch auch Partner, die keine datenabhängige Transition enthalten, Abhängigkeiten zwischen Datenwerten berücksichtigen müssen. Auch dies steht, wie im vorangehenden Beispiel, in engem Zusammenhang zur Werterzeugung.



(a) N'' (b) $S_1'' \in Partner(N'')$ (c) $S_2'' \notin Partner(N'')$ (d) $S_3'' \in Partner(N'')$

Abbildung 5: Ein offenes Netz N'' und kompatible offene Netze

Beispiel 3 Das Netz N'' aus Abb. 5 empfängt zunächst zwei Werte auf a_1 und a_2 und trifft anschließend eine nichtdeterministische Entscheidung: Es wird eine Marke entwe-

der (a) auf p_5 oder (b) auf p_6 produziert. Diese Wahl wird der Umgebung kommuniziert, indem entweder (a) der auf a_1 empfangene Wert oder (b) der auf a_2 empfangene Wert auf b gesendet wird. N'' erwartet anschließend (a) eine Nachricht auf c_1 oder (b) eine Nachricht auf c_2. Ein Partner von N'' darf niemals zwei gleiche Werte auf a_1 und a_2 senden. In diesem Fall sind die beiden Fälle (a) und (b) anhand der auf b empfangenen Nachricht nicht unterscheidbar und der Partner kann nicht wissen, ob N'' eine Antwort auf c_1 oder c_2 erwartet. Der Guard $[x \neq y]$ an Transition s_1 in S_3'' ist also zwingend erforderlich, um zu verhindern, dass N'' möglicherweise einen Deadlock erreicht. Aufgrund des Guards ist s_1 eine datenabhängige Transition. Ebenfalls datenabhängig sind jeweils die Transitionen s_2 und s_3 in S_1'' sowie s_3, s_4 in S_3'', welche eine Fallunterscheidung abhängig vom auf b empfangenen Wert treffen. Wir gelangen zu der folgenden Vermutung:

Vermutung 2. Jeder Partner von N'' enthält mindestens eine datenabhängige Transition.

Diese Folgerung ist bemerkenswert, da N'' selbst weder datenabhängige noch erzeugende Transitionen enthält. Die Partner verwenden also Ausdrucksmittel, die in N'' nicht vorkommen: S_1'' verwendet Konstanten, S_3'' einen Guard. N'' verwendet weder Konstanten noch Guards.

3 Schlussfolgerungen und Ausblick

Die untersuchten Beispiele zeigen, dass selbst bei beschränkten Ausdrucksmitteln (nur Variablen, keine Guards) vergleichsweise komplexe Anforderungen an Partner eines offenen Netzes entstehen. Insbesondere zeigt Beispiel 3, dass jeder Partner eines offenen Netzes unter Umständen Ausdrucksmittel verwendet, die im offenen Netz selbst nicht vorkommen. Dies ist vor allem für das Problem der Partnersynthese wichtig. Offen ist, welche Typen von Transitionen hinreichen, um jeden Partner eines offenen Netzes, dass keine datenabhängigen Transitionen enthält, zu beschreiben. Ebenfalls offen ist, welche Techniken zum Beweis der aufgestellten Vermutungen notwendig sind.

Literatur

1. Lohmann, N., Massuthe, P., Wolf, K.: Operating guidelines for finite-state services. In: Kleijn, J., Yakovlev, A. (eds.) 28th International Conference on Applications and Theory of Petri Nets and Other Models of Concurrency, ICATPN 2007, Siedlce, Poland, June 25-29, 2007, Proceedings. Lecture Notes in Computer Science, vol. 4546, pp. 321–341. Springer-Verlag (2007)
2. Massuthe, P., Serebrenik, A., Sidorova, N., Wolf, K.: Can I find a partner? Undecidablity of partner existence for open nets. Information Processing Letters 108(6), 374–378 (November 2008)
3. Weinberg, D.: Efficient controllability analysis of open nets. In: Bruni, R., Wolf, K. (eds.) Web Services and Formal Methods, Fifth International Workshop, WS-FM 2008, Milan, Italy, September 4–5, 2008, Proceedings. Lecture Notes in Computer Science, Springer-Verlag (September 2008)