Kurzfassung

Model Checking ist ein Ansatz zur Validierung der Korrekheit eines Hard- oder Softwaresystems. Dazu wird das System durch ein formales Modell beschrieben und Systemeigenschaften werden meist in temporaler Logik spezifiziert. Ein Model Checker untersucht dann vollautomatisch, ob das Modell eine Eigenschaft erfüllt, indem er dessen Zustandsraum untersucht. Da jedoch die Anzahl der Zustände exponentiell mit der Größe des Systems wachsen kann –was als *Zustandsraumexplosion* bezeichnet wird– ist die Entwicklung und Anwendung von Methoden unumgänglich, die es ermöglichen beim Model Checking mit Systemen umzugehen, die einen großen Zustandsraum haben.

Ein etablierter Formalismus zur Beschreibung von asynchronen Systemen, die durch Nebenläufigkeit, Parallelität und Nichtdeterminismus gekennzeichnet sind, sind Petri-Netze. Der Petri-Netz-Formalismus bietet eine Fülle von Analysetechniken und eine intuitive graphische Darstellung.

In der vorliegenden Arbeit werden zwei Reduktionsansätze für Petri-Netze vorgestellt, *Petri-Netz Slicing* und *Cutvertex Reduktionen*. Beide Ansätze zielen darauf ab, der Zustandsraumexplosion beim Model Checken entgegenzuwirken. Dazu transformieren sie ein gegebenes Petri-Netz in ein kleineres Netz, so dass gleichzeitig die untersuchte Eigenschaft bewahrt wird. Da Petri-Netz-Reduktionen das Modell transformieren, können sie leicht mit anderen Methoden kombiniert werden.

Die Kernidee beider Reduktionsansätze ist, dass temporal-logische Eigenschaften sich meist auf nur wenige Stellen eines Petri-Netzes beziehen und daher häufig Teile eines Petri-Netzes identifiziert werden können, die die untersuchte Eigenschaft nicht oder nur unwesentlich beeinflussen. Wir nennen die Menge der Petri-Netzstellen auf die sich eine temporal-logische Formel φ bezieht $scope(\varphi)$. Für ein gegebenes Netz Σ und eine temporal-logische Eigenschaft φ bestimmen beide Ansätze ein Netz Σ', das wenigstens $scope(\varphi)$ enthält, und vereinfachen das übrige Netz so, dass Σ' in Bezug auf φ äquivalent zu Σ ist.

Wir zeigen, dass es genügt, eine schwache Form von Fairness anzunehmen,

die wir *relative Fairness* nennnen, um Lebendigkeitseigenschaften zu erhalten.

Petri-Netz Slicing, ein durch Program Slicing inspirierter Ansatz, bestimmt ein reduziertes Netz beginnend von $scope(\varphi)$, indem das Netz um relevante Transitionen und deren Eingabestellen iterativ erweitert wird. Wir formulieren zwei solcher Slicingalgorithmen, *$CTL^*_{-\mathsf{X}}$ Slicing* und *Safety Slicing*, und zeigen, dass die so reduzierten Netze Falsifikation von $\forall$CTL*-Eigenschaften erlauben. Wir zeigen weiterhin, dass $\text{CTL}^*_{-\mathsf{X}}$ Slicing $\text{CTL}^*_{-\mathsf{X}}$-Eigenschaften bewahrt, wenn relative Fairness für Σ angenommen wird. Das üblicherweise aggressivere Safety Slicing bewahrt stotter-invariante Sicherheitseigenschaften.

Cutvertex Reduktionen sind ein dekompositioneller Ansatz. Ein monolithisches Petri-Netz wird in einen Kernel, der $scope(\varphi)$ enthält, und Umgebungsnetze zerlegt. Die Umgebungsnetze werden durch eines von sechs vorgegebenen, sehr kleinen *Summarynetzen* ersetzt. Um das geeignete Summarynetz zu identifizieren, wird ein Umgebungsnetz isoliert vom Gesamtsystem durch Model Checking untersucht. Dieser Identifikationsschritt wird durch unsere strukturellen *Pre/Postset Optimierungen* beschleunigt. Wir führen außerdem Mikroreduktionen ein, die die kleinsten Umgebungen direkt, das heißt ohne Untersuchung durch einen Model Checker, ersetzen. Wir zeigen, dass unter relativer Fairness Cutvertex Reduktionen alle Eigenschaften erhält, die in $\text{LTL}_{-\mathsf{X}}$ formulierbar sind.