Anleitung zur Implementierung von Funktionspartitionierung/-mapping mit Hilfe von Constraint Programming

Johannes Schlatow

TU Braunschweig, IDA

schlatow@ida.ing.tu-bs.de

21. März 2019

draft - 21. März 2019

Zusammenfassung

Lorem ipsum.

---

1 Einleitung und Überblick

Aktueller Stand

Der *Multi-Change Controller* sucht auf Basis einer vorgegebenen Funktionsarchitektur (*function architecture*) eine Systemkonfiguration, bestehend aus Softwarekomponenten. Die Funktionsarchitektur wird als Graph gespeichert und visualisiert. Dabei stellen die Knoten die Funktionsblöcke darstellen und die Kanten die (funktionalen) Abhängigkeiten Derzeit ist jeder Knoten bereits einer Plattformkomponente (z.B. Prozessor, Subsystem) zugewiesen (*mapping*) bevor der MCC mit seiner Suche nach einer Konfiguration beginnt.

Ziel

Der MCC soll um die Lösung des Mapping-Problems erweitert werden, d.h. gegeben eine Funktionsarchitektur sowie eine Zielplattform, soll der MCC eine Zuweisung von Funktionsblöcken auf Plattformkomponenten finden.

Methodik

Das Mapping-Problem ist eine bekanntes *Constraint Satisfaction Problem* (CSP). Es existieren diverse Methoden zur Lösung diese Problems mit unterschiedlichen Constraints und ggf. Optimierungszielen. Dazu zählen *satisfiability modulo theories* (SMT), *(integer) linear programming* (ILP) sowie *constraint programming* (CP).

An dieser Stelle soll CP angewandt werden.

Eine praktische Einführung ähnlicher Probleme wird auf der Website der Google OR Tools gegeben.

Einschränkung

Das Mapping-Problem beschreibt nur einen Teil der Requirements und Constraints, die im weiteren Verlauf vom MCC berücksichtigt werden müssen. Wir gehen also davon aus, dass das Mapping-Problem nicht vollständig den Lösungsraum einschränkt, d.h. die formulierten Constraints sollen den Charakter von *notwendigen* Bedingungen haben.

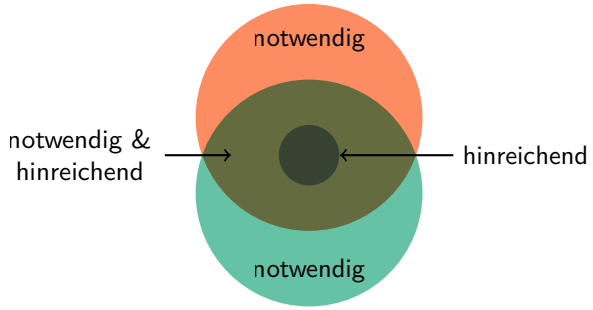

Abbildung 1: Notwendige vs. hinreichende Bedingungen

2 Constraints und Zielfunktion

Gegeben:

- Graph $G = (V, E)$ mit Knoten V und Kanten E.
- Menge an Plattformkomponenten P.
- Kompatibilitätsfunktion $C(v \in V) \subseteq P$.

Gesucht:

- Mapping $f : V \to P$

Constraints:

- Verfügbarer Speicher pro $p \in P$ und Speicherkosten pro $v \in V$ (abhängig von Mapping, d.h. $f : V \times P \to \mathbb{N}$)
  - Die Summe der Speicherkosten der auf $p \in P$ gemappten $v \in V$ darf den verfügbaren Speicher nicht überschreiten.

- Utilisation pro $v \in V$ (abhängig von Mapping)
  - Die Last/Utilisation der auf $p \in P$ gemappten $v \in V$ darf 100% nicht überschreiten.
- Partitioning Constraints
  - Es können Knotenpaare (u, v) definiert werden, die auf der gleichen Ressource laufen müssen.
  - Es können Knotenpaare (u, v) definiert werden, die auf der unterschiedlichen Ressourcen laufen müssen.
- Size Constraints
  - Es können Mengen $g \subseteq V$ definiert werden, die nicht zusammen auf eine bestimmte Ressource passen.

Zielfunktion:

1. Minimierung der Partitionierungskosten
   - Für jedes Knotenpaar (u, v) können Kosten (Penalty) definiert werden, wenn sie auf der gleichen Ressource laufen.
2. Minimierung der Kommunikationskosten
   - Für jede Kante $e \in E$ fallen Kommunikationskosten an, wenn sie auf unterschiedlichen Ressourcen laufen.
3. Minimierung der Kommunikationslatenz
   - Für jede Kante $e \in E$ können Kommunikationszeiten definiert werden, wenn sie auf unterschiedlichen Ressourcen laufen.

Die Reihenfolge bzw. Priorität der Zielfunktionkomponenten soll konfigurierbar sein.

3 Umsetzung

3.1 Schritt 0

Johannes

Die Funktionsarchitektur ist gegeben als Graph. Eine AnalyseEngine wird implementiert, welche eine `map` Operation für den Parameter *mapping* durchführt. Dabei werden jedem Funktionsblock eine Menge an kompatiblen Plattformkomponenten zugewiesen ($C(v)$).

3.2 Schritt 1

Edgard

Anschließend wird über eine `assign` Operation jedem Knoten eine Ressource zugewiesen. Hierzu werden Constraints und Zielfunktionen (für die Optimierung) als CP formuliert und mittels Google OR Tools gelöst. Wichtig ist hierbei, dass die Kosten und Constraints nicht unbedingt vollständig bekannt sind, d.h. die Implementierung muss mit fehlenden Werten umgehen können.

3.3 Schritt 2

Edgard

Im weiteren Verlauf der Suche durch den MCC kann sich eine gefundenes Mapping als nicht umsetzbar herausstellen. Hierzu ist es erforderlich die während der Suche gefundenen Partitionierungs- oder Size-Constraints als Zustand zu speichern. Wir nehmen momentan an, dass diese Constraints für die gegeben Repositories und die gegebene Zielplattform allgemein gültig sind.

4 Nützliches

- Google OR Tools für Archlinux
- Google OR Tools Python Library für Archlinux