14.2 Analytische Semantik von Programmen

Ziel:
- Schwache Speichermodelle definieren
- Semantik von Programmen unter solchen Speichermodellen definieren

Idee: Schwache Speichermodelle sollen Kommunikation einschränken, nicht aber Kontroll- und Datenfluss (nur indirekt)

Dazu:
- Kommunikation findet über Memory-Events statt.
  - ↳ Anarchische Semantik enthält viele unnötige Events
  - ↳ Abstrahieren!

Beobachtung:

Wenn $\pi = (P, rf)$ eine Execution ist, dann können wir diese nur anhand ihrer Memory-Events $E = Memory(P)$ und ihrer rf-Relation rekonstruieren:

- Die initiale Konfig. per Thread ist eindeutig
- Im Falle von lokalen Zuweisungen, Branching und Writes ist das erzeugte Event ~~eindeutig~~ und der Nachfolgezustand eindeutig (bis auf Timestamp) durch die aktuelle Konfig. definiert
- Für die Timestamps wählen wir beliebige entlang der totalen Ordnung
- Im Falle von Reads, übernehmen wir den entsprechenden Read aus E
- Am Ende verbinden wir unsere Writes und Reads nach dem gegebenen rf.

↳ Die rekonstruierte Execution stimmt mit π bis auf Umbenennung der Timestamps überein!

Theorem 1: Die Mengen $Exec(P)$ und $\{(Memory(P), rf) \mid (P, rf) \in Exec(P)\}$ sind isomorph für jedes Programm P.

Beweis: Formalisiere Beobachtung □

Theorem 1 zeigt, dass die Kommunikation rf bereits Kontroll- und Datenfluss bestimmt.

↳ Speichermodelle müssen nur Kommunikation beeinflussen!

Def (Candidate Execution):

Sei $\pi = (\rho, rf)$ eine Execution eines Programms P.

Die dazugehörige Candidate Execution ist ein Graph $G(\pi) = (E, po, rf, \ldots)$ bestehend aus:

- der Menge $E = Memory(\rho)$ (eventuell weitere wie Fences)
- der Programmordnung $po \subseteq E \times E$:

  $(e_1, e_2) \in po \iff (e_1 \in IW \wedge e_2 \notin IW)$

  ODER $(thr(e_1) = thr(e_2) \wedge \Theta(e_1) < \Theta(e_2))$
- der Read-From Relation $rf \subseteq Write(E) \times Read(E)$
- potenziell weiteren wie Kontroll-, Daten- und Addressabhängigkeiten.

Die Menge aller Candidate Executions (von Programm P) bezeichnen wir mit CExec (CExec(P))

Def (schwaches Speichermodell):

Ein schwaches Speichermodell ist eine Funktion $MM : CExec \to \mathbb{B}$.

Wir schreiben $G \models MM$, falls $MM(G) = true$.

Die Semantik von MM ist definiert als

$$[\![MM]\!] := \{\pi \in Exec \mid G(\pi) \models MM\} \subseteq Exec.$$

Def: (Analytische Semantik):

Die analytische Semantik von P unter MM ist

$$[\![P]\!]_{MM} := [\![P]\!] \cap [\![MM]\!]$$

Nun wissen wir wie Programme unter schwachen Speichermodellen funktionieren.
Wir wissen auch wie wir Programme formulieren.
Wir wissen aber noch nicht wie wir schwache Speichermodelle formulieren

↳ Mehr dazu im nächsten Kapitel !?