A10.1

```
void enqueue(int x) {
    Node* node = new Node();
    node->data = x;
    node->next = NULL;
    while (true) {
        Node* tail = Tail;
        Node* next = tail->next;
        if (tail != Tail) continue;
        if (next != NULL) {
            CAS(Tail, tail, next);
            continue;
        }
        if (CAS(tail->next, next, node)) {
            CAS(Tail, tail, node);
            break;
        }
    }
}
```

„Helping", ändert logischen Queue-Inhalt nicht

```
bool b = false;
atomic {
    if (tail->next == next) {
    // LP
    tail->next = node;
        b = true
    }
}
if (b) {...
```

Erfolg entscheidet nicht über Ausgang der Methode

```
(bool, int) dequeue() {
  while (true) {
    Node * head = Head;
    Node * tail = Tail;
    Node * next = head->next;
    if (head != Head) continue;
    if (next == NULL) {
      return (false, 0);
    }
    if (head == tail) {
      CAS(Tail, tail, next);
      continue;
    }
    int x = next->data;
    if (CAS(Head, head, next)) {
      return (true, x);
    }
  }
}
```

(auch *) → "Prophecy"

falls next == NULL

```
if (havoc) {
    head = Head;
    tail = Tail;
    next = head->next; // LP
    assume (head == Head);
} else {
    head = Head;
    tail = Tail;
    next = head->next; // kein LP
    assume (head != Head);
    continue;
}
```

LP im "Head == head" Fall,
wie in enqueue

A 10.2

$$\frac{b = \varepsilon}{(h, lcf[i \mapsto (\mathit{mfence}; c, s, b)]) \xrightarrow{(i, \mathit{mfence})} (h, lcf[i \mapsto (c, s, b)])} \ (\text{MFENCE})$$

A 10.3

Sei $\tau.act \in [\![P]\!]_{TSO}$. Dann auch $\tau \in [\![P]\!]_{TSO}$.

Sei HBTrace (τ) kreisfrei, HBTrace (τ. act) nicht kreisfrei.

ZZ: act ist ein Store

Dazu zeige: andere Anweisungen erzeugen keine Kreise.

Nach Def: $\text{HBTrace}(\tau.act) = (N \cup \{n\}, \lambda', \rightarrow_{po}', \rightarrow_{co}', \rightarrow_{rf}')$

für $\text{HBTrace}(\tau) = (N, \rightarrow_{po}, \rightarrow_{co}, \rightarrow_{rf})$

- Im „Nicht-Store" Fall fügen wir neuen Knoten hinzu,

also: $n \notin N$, $\lambda' = \lambda \cup \{(n, act)\}$

$\rightarrow_{po}' := \rightarrow_{po} \cup \{(\max(\rightarrow_{po}), n)\}$

// eingehende Kante für n, keine ausgehende Kante

- $\rightarrow_{co}' = \rightarrow_{co}$
- entweder $\rightarrow_{rf}' = \rightarrow_{rf}$, falls act kein Load

oder $\rightarrow_{rf}' = \rightarrow_{rf} \cup \{(\max(\rightarrow_{co}^{adr}), n)\}$,

falls $act = (t, ld, adr, val)$

// fügen höchstens eingehende Kanten hinzu

Damit wurde kein Kreis eingeführt und es muss sich bei act um ein Store handeln.

Ferner gilt:

$\to'_{hb}$ ist kreisfrei falls $\to_{hb}$ kreisfrei. (im „Nicht-Store" Fall)

Betrachte From-Read-Relation.

Falls act kein Load, dann $\to'_{fr} = \to_{fr}$.

Falls act $= (t, ld, adr, val)$, so ist

$$\to'_{fr} = \to_{fr} \cup \left\{ (n, n') \;\middle|\; \begin{array}{l} \lambda(n') = (\_, st, adr, \_) \\ \land\, \exists n''.\ \lambda(n'') = (\_, st, adr, \_) \\ \land\, n'' \to_{rf} n \land n'' \to_{co} n' \end{array} \right\}$$

Wissen: einzige $\to_{rf}$ Kante nach n ist maximal bzgl. $\to_{co}$.
Also existiert kein solches n'' und wir kriegen keine $\to_{fr}$ Kanten.

$$\to'_{fr} = \to_{fr}$$

Also ist $\to'_{hb}$ kreisfrei falls $\to_{hb}$ kreisfrei

$\blacksquare$

$\boxed{10.4}$ Sei $\mathrm{Traces}_{SC}(P)$ DRF.

Angenommen $\mathrm{Traces}_{TSO}(P) \not\subseteq \mathrm{Traces}_{SC}(P)$.

Dann existiert ein kürzestes $\sigma.act \in [\![P]\!]_{TSO}$ mit:

- $\mathrm{HBTrace}(\sigma.act) \notin \mathrm{Traces}_{SC}(P)$ und
- ~~HBTraces (σ.act) hat minimale Anzahl Knoten~~.

Nach Minimalität: $\mathrm{HBTrace}(\sigma) \in \mathrm{Traces}_{SC}(P)$.

Nach Shasha & Snir:

- $\rightarrow_{hb}$ azyklisch // happens-before für σ
- $\rightarrow'_{hb}$ zyklisch // happens-before für $\sigma.act$

Nach 10.3 ist act nun ein Store, $act = (t, st, adr, val)$.

Sei $\mathrm{HBTrace}(\sigma) = (N, \lambda, \rightarrow_{po}, \rightarrow_{co}, \rightarrow_{rf})$ mit $\rightarrow_{fr}$,

$\mathrm{HBTrace}(\sigma.act) = (N, \lambda', \rightarrow'_{po}, \rightarrow'_{co}, \rightarrow'_{rf})$ mit $\rightarrow'_{fr}$.

Da act Store ist, wird N nicht erweitert.

Sei n Knoten für act, also $\lambda'(n) = act$ $(\lambda(n) = (t, isu))$.

Kreis in $\rightarrow'_{hb}$ enthält mindestens 2 Threads, hat Form

$$n' \xrightarrow{*}_{hb'} n'' \xrightarrow{*}_{po'} n \xrightarrow{*}_{hb'} n'$$

mit:

$n' \in N \setminus \{n\}$ $\qquad \lambda'(n') = (t', ld/st, adr', val')$

$n'' \in N \setminus \{n'\}$ $\qquad \lambda'(n'') = (t, ld/st, adr'', val'')$

$t' \neq t$

Betrachte $n' \xrightarrow{hb'} n''$.

Wir haben $n' \xrightarrow{rf' \cup fr' \cup co'} n''$ da n', n'' von unterschiedlichen Threads stammen.

Nach Def rf', co', fr' sind n', n'' Actions auf der selben Adresse, also $adr' = adr''$. Ferner können nicht beides Loads sein.

Damit ~~ist~~ n', n'' ein Data Race.

Beachte, dass $n'' = n$ oder $n'' \neq max(\xrightarrow{t}_{po})$.

Damit liefert Ann. ein Data Race in SC. ↯

∎