Project Formele Systeemmodellering voor Software

Bruno Corijn, Jasper Vander Jeugt, Toon Willems

6 januari 2013

1 Deel 1: ACL2

Voor de priority-queue zijn er drie versies terug te vinden. Oorspronkelijk waren we begonnen met de queue te implementeren als een binomial heap. Dit heeft als voordeel dat het veel sneller is dan de andere alternatieven. Het grote nadeel was echter de moeilijke formele verificatie van deze datastructuur.

Hierna hebben we geprobeerd de queue te implementeren als eenvoudige linked list. De eigenschappen hiervan zijn min of meer tegenovergesteld aan die van de binomial heap: de correctheid was makkelijker te bewijzen maar het was echter ook zeer inefficiënt voor de gebruiker, waardoor we ook dit niet verder hebben uitgewerkt.

Uiteindelijk hebben we gekozen voor een binaire boom. In het slechtste geval zal ook deze datastructuur traag werken, gezien er niet wordt geherbalanceerd maar gemiddeld zal deze implementatie sneller zijn dan de linked list.

De finale implementatie kan teruggevonden worden in `binary-tree.lisp`. De twee andere (onafgewerkte) implementaties bevinden zich in `linked-list.lisp` en `binomial-heap.lisp`.

1.1 Interface

We bespreken eerst kort de "publieke API"van de queue. Nieuwe wachtrijen aanmaken:

- `queue-empty` maakt een lege wachtrij aan.
- `queue-singleton k v` maakt een wachtrij aan met een element met als waarde `v` en als prioriteit `k`.

Eigenschappen van queues opvragen:

- `queue-null queue` kijkt of de gegeven wachtrij leeg is.
- `queue-size queue` bepaalt het aantal elementen in de wachtrij.
- `queue-find-min-value queue` geeft de waarde terug die bij de hoogste prioriteit hoort.

Wachtrijen bewerken:

- `queue-insert k v queue` voegt een element met als waarde `v` en als prioriteit `k` toe aan de wachtrij.
- `queue-delete-min queue` verwijdert het element met de hoogste prioriteit.
- `queue-merge q1 q2` voegt de wachtrijen `q1` en `q2` tesamen.
- `queue-change-priority k v queue` verandert de prioriteit van de elementen met als waarde `v` naar `k`.

1.2 Formele specificaties

De specificaties zijn terug te vinden als theorems, die gebruik maken van enkele hulp functies.

Deze functies controleren de volgende zaken: of alle elementen in de queue een kleinere prioriteit hebben dan een gegeven `x`, of alle elementen in de queue een grotere of gelijke prioriteit hebben aan een gegeven `x`, of er een element met prioriteit `k` en value `v` terug te vinden is in een gegeven queue.

Het belangrijkste hulppredicaat is `queue-valid`, dit gaat na of de ordening van de gehele boom correct is. Dit is dan ook een nodige voorwaarde bij de meeste theorema's.

We specifiëren dat na enkele mogelijke update (een element toevoegen, verwijderen...) `queue-valid` behouden blijft. Op die manier kunnen we zeggen dat `queue-valid` altijd klopt als de wachtrij is opgebouwd via de functies uit de publieke API.

TODO. In de code is er bij elk theorema commentaar voorzien zodat het duidelijk zou moeten zijn waartoe het onderstaand theorema dient.

1.3 Verificatie

De output van de automatische verificatie van de implementatie als binaire boom is te vinden als bijlage aan ons verslag.

2 Deel2: TLA/TLC