A Persistent Functional Language for Concurrent Transaction Processing

Lesley Wevers

dr. Marieke Huisman
dr.ir. Ander de Keijzer
prof.dr. Jaco van de Pol

28 Augustus, 2012

Voorbeeld

Bank

- A heeft 100 euro
- B heeft 100 euro

Voorbeeld

Bank

- A heeft 100 euro
- B heeft 100 euro

Transactie

A maakt 40 euro over naar B:

- A: -40 euro
- B: $+40$ euro

Voorbeeld

Bank

- A heeft 100 euro
- B heeft 100 euro

Transactie

A maakt 40 euro over naar B:

- A: -40 euro
- B: $+40$ euro

Voorbeeld

Bank

- A heeft 60 euro
- B heeft 100 euro

Transactie

A maakt 40 euro over naar B:

- A: -40 euro
- B: $+40$ euro

Voorbeeld

Bank

- A heeft 60 euro
- B heeft 100 euro

Transactie

A maakt 40 euro over naar B:

- A: -40 euro
- B: $+40$ euro

A problem has been detected and Windows has been shut down to prevent damage
to your computer.

DRIVER_IRQL_NOT_LESS_OR_EQUAL

If this is the first time you've seen this Stop error screen,
restart your computer, If this screen appears again, follow
these steps:

Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any Windows updates you might need.

If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.

Technical information:

*** STOP: 0x000000D1 (0x0000000C,0x00000002,0x00000000,0xF86B5A89)

***   gv3.sys - Address F86B5A89 base at F86B5000, DateStamp 3dd991eb

Beginning dump of physical memory
Physical memory dump complete.
Contact your system administrator or technical support group for further
assistance.

Voorbeeld

Bank

- A heeft 60 euro
- B heeft 100 euro

Voorbeeld

Bank

- A heeft 100 euro
- B heeft 100 euro

Voorbeeld

Bank

- A heeft 100 euro
- B heeft 100 euro

Transactie

A maakt 40 euro over naar B:

- A: -40 euro
- B: $+40$ euro

Voorbeeld

Bank

- A heeft 100 euro
- B heeft 100 euro

Transactie

A maakt 40 euro over naar B:

- A: -40 euro
- B: $+40$ euro

Transactie

Hoeveel geld is er in totaal?

- Totaal: ...

Voorbeeld

Bank

- A heeft 100 euro
- B heeft 100 euro

Transactie

A maakt 40 euro over naar B:

- A: -40 euro
- B: $+40$ euro

Transactie

Hoeveel geld is er in totaal?

- Totaal: ...

Voorbeeld

Bank

- A heeft 100 euro
- B heeft 100 euro

Transactie

A maakt 40 euro over naar B:

- A: -40 euro
- B: $+40$ euro

Transactie

Hoeveel geld is er in totaal?

- Totaal: $100 + \ldots$

Voorbeeld

Bank

- A heeft 100 euro
- B heeft 100 euro

Transactie

A maakt 40 euro over naar B:

- A: -40 euro
- B: $+40$ euro

Transactie

Hoeveel geld is er in totaal?

- Totaal: $100 + \ldots$

Voorbeeld

Bank

- A heeft 60 euro
- B heeft 100 euro

Transactie

A maakt 40 euro over naar B:

- A: -40 euro
- B: $+40$ euro

Transactie

Hoeveel geld is er in totaal?

- Totaal: $100 + \ldots$

Voorbeeld

Bank

- A heeft 60 euro
- B heeft 100 euro

Transactie

A maakt 40 euro over naar B:

- A: -40 euro
- B: $+40$ euro

Transactie

Hoeveel geld is er in totaal?

- Totaal: $100 + \ldots$

Voorbeeld

Bank

- A heeft 60 euro
- B heeft 140 euro

Transactie

A maakt 40 euro over naar B:

- A: -40 euro
- B: $+40$ euro

Transactie

Hoeveel geld is er in totaal?

- Totaal: $100 + \ldots$

Voorbeeld

Bank

- A heeft 60 euro
- B heeft 140 euro

Transactie

A maakt 40 euro over naar B:

- A: -40 euro
- B: $+40$ euro

Transactie

Hoeveel geld is er in totaal?

- Totaal: $100 + \ldots$

Voorbeeld

Bank

- A heeft 60 euro
- B heeft 140 euro

Transactie

A maakt 40 euro over naar B:

- A: -40 euro
- B: $+40$ euro

Transactie

Hoeveel geld is er in totaal?

- Totaal: $100 + 140$

Voorbeeld

Bank

- A heeft 60 euro
- B heeft 140 euro

Transactie

A maakt 40 euro over naar B:

- A: -40 euro
- B: $+40$ euro

Transactie

Hoeveel geld is er in totaal?

- Totaal: 240

Transacties

Transactie

- Een systeem heeft een *toestand*.

Transacties

Transactie

- Een systeem heeft een *toestand*.
- Een *transactie* is een verzameling *operaties* op een toestand.

Transacties

Transactie

- Een systeem heeft een *toestand*.
- Een *transactie* is een verzameling *operaties* op een toestand.

ACID eigenschappen voor transacties

Atomic: Alle operaties worden uitgevoerd, of geen enkele operatie wordt uitgevoerd.

Transacties

Transactie

- Een systeem heeft een *toestand*.
- Een *transactie* is een verzameling *operaties* op een toestand.

ACID eigenschappen voor transacties

Atomic: Alle operaties worden uitgevoerd, of geen enkele operatie wordt uitgevoerd.

Consistent: Na elke transactie is het systeem in een correcte toestand.

Transacties

Transactie

- Een systeem heeft een *toestand*.
- Een *transactie* is een verzameling *operaties* op een toestand.

ACID eigenschappen voor transacties

Atomic: Alle operaties worden uitgevoerd, of geen enkele operatie wordt uitgevoerd.

Consistent: Na elke transactie is het systeem in een correcte toestand.

Isolated: Het lijkt als of de transacties één voor één worden uitgevoerd.

Transacties

Transactie

- Een systeem heeft een *toestand*.
- Een *transactie* is een verzameling *operaties* op een toestand.

ACID eigenschappen voor transacties

Atomic: Alle operaties worden uitgevoerd, of geen enkele operatie wordt uitgevoerd.

Consistent: Na elke transactie is het systeem in een correcte toestand.

Isolated: Het lijkt als of de transacties één voor één worden uitgevoerd.

Durable: Het effect van een transactie is blijvend.

Transactieverwerkingssystemen

Transactieverwerkingssystemen

- Banken

Transactieverwerkingssystemen

Transactieverwerkingssystemen

- Banken
- Ticket reservering

Transactieverwerkingssystemen

Transactieverwerkingssystemen

- Banken
- Ticket reservering
- Inventarisatie systemen

Transactieverwerkingssystemen

Transactieverwerkingssystemen

- Banken
- Ticket reservering
- Inventarisatie systemen
- Veilingen

Transactieverwerkingssystemen

Transactieverwerkingssystemen

- Banken
- Ticket reservering
- Inventarisatie systemen
- Veilingen
- Internetwinkels

Transactieverwerkingssystemen

Transactieverwerkingssystemen

- Banken
- Ticket reservering
- Inventarisatie systemen
- Veilingen
- Internetwinkels

Uitdaging

Duizenden gelijktijdige gebruikers:

Transactieverwerkingssystemen

Transactieverwerkingssystemen

- Banken
- Ticket reservering
- Inventarisatie systemen
- Veilingen
- Internetwinkels

Uitdaging

Duizenden gelijktijdige gebruikers:

- Alles moet goed gaan. (ACID eigenschappen)

Transactieverwerkingssystemen

Transactieverwerkingssystemen

- Banken
- Ticket reservering
- Inventarisatie systemen
- Veilingen
- Internetwinkels

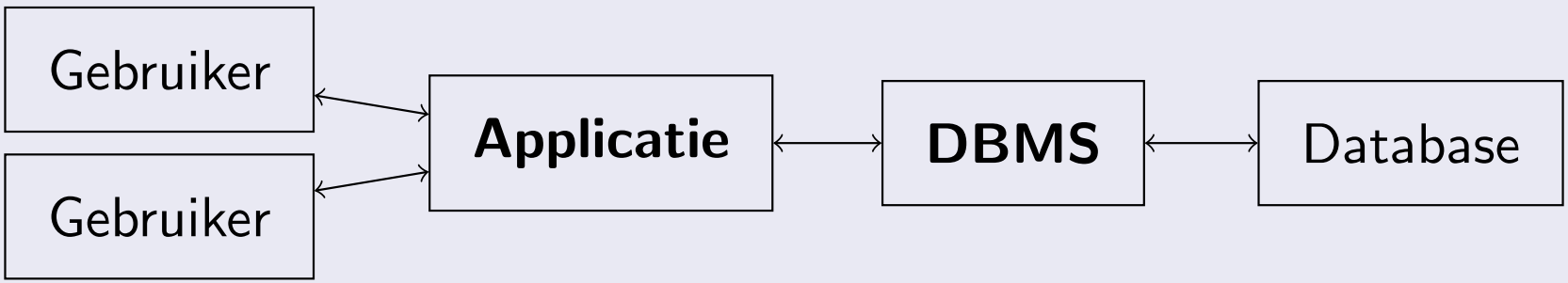
Uitdaging

Duizenden gelijktijdige gebruikers:

- Alles moet goed gaan. (ACID eigenschappen)
- Iedereen wil snel antwoord.

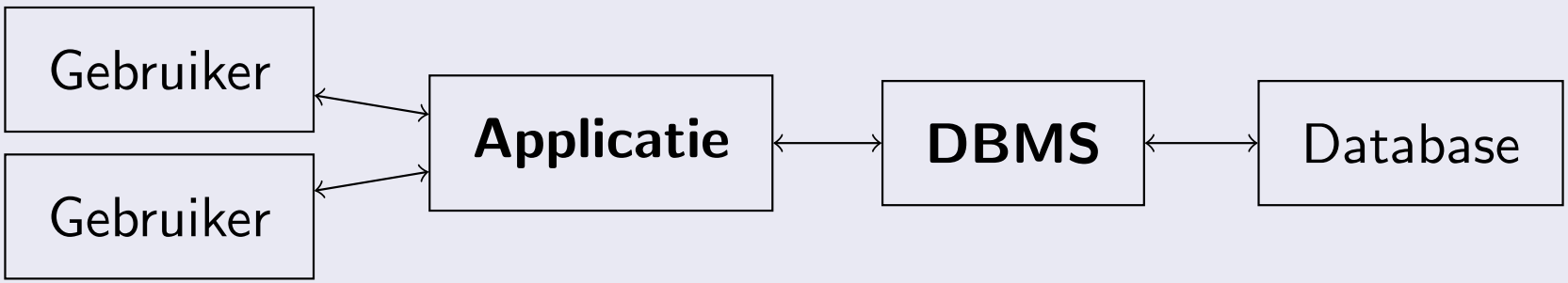
Traditioneel Model

Traditionele Architectuur



Traditioneel Model

Traditionele Architectuur

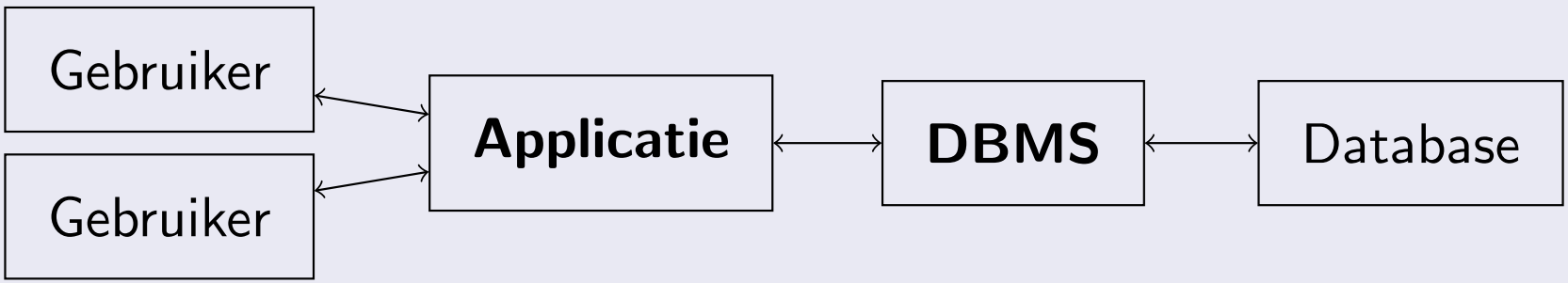


Limitaties

- Most DBMS's only support isolation partially.

Traditioneel Model

Traditionele Architectuur

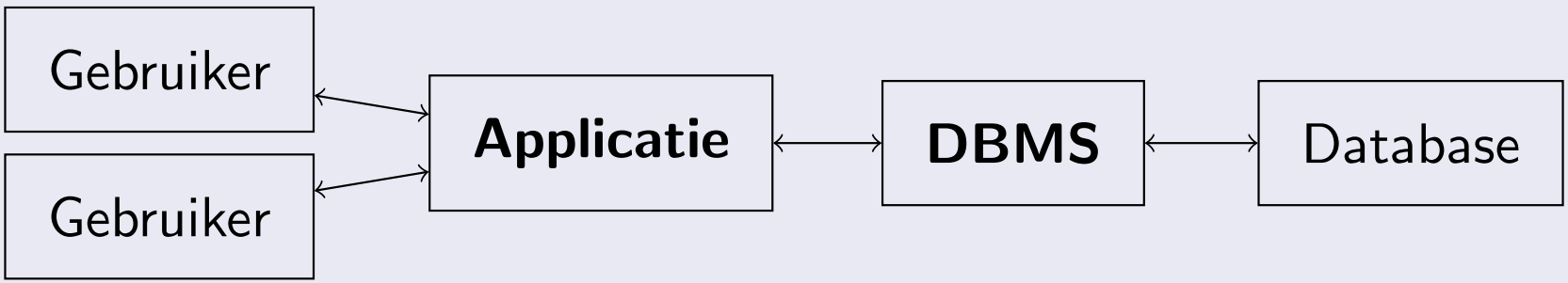


Limitaties

- Most DBMS's only support isolation partially.
- Application and DBMS have different type system.

Traditioneel Model

Traditionele Architectuur

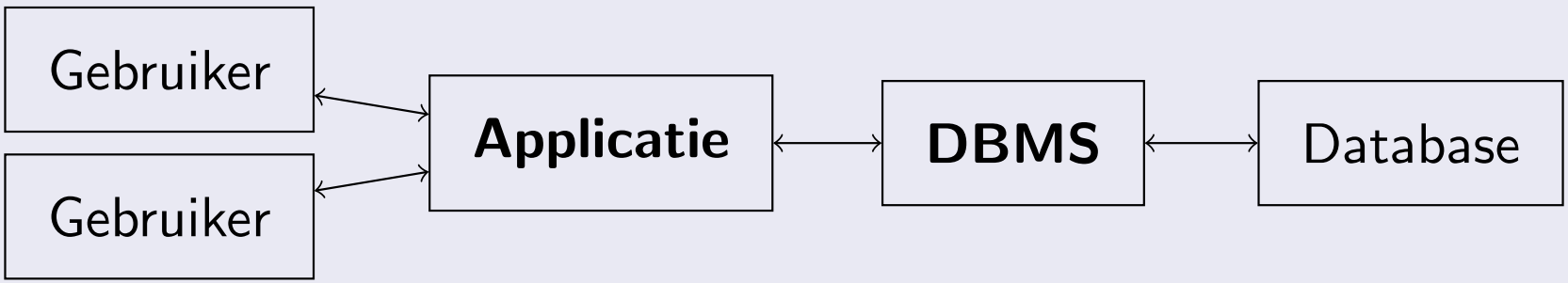


Limitaties

- Most DBMS's only support isolation partially.
- Application and DBMS have different type system.
- Serial interface between application and DBMS.

Traditioneel Model

Traditionele Architectuur

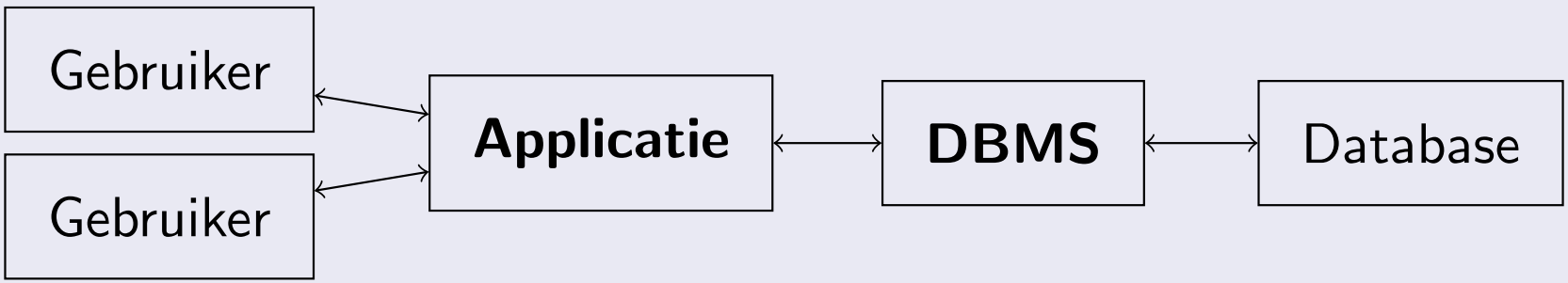


Limitaties

- Most DBMS's only support isolation partially.
- Application and DBMS have different type system.
- Serial interface between application and DBMS.
- Distributed system complicates implementation.

Traditioneel Model

Traditionele Architectuur

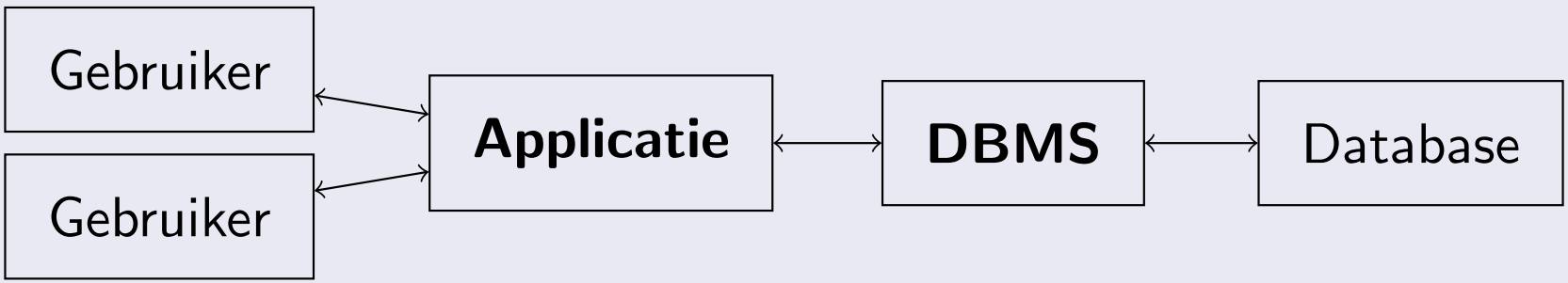


Limitaties

- Most DBMS's only support isolation partially.
- Application and DBMS have different type system.
- Serial interface between application and DBMS.
- Distributed system complicates implementation.
- DBMS's are vulnerable to command injection attacks.

Traditioneel Model

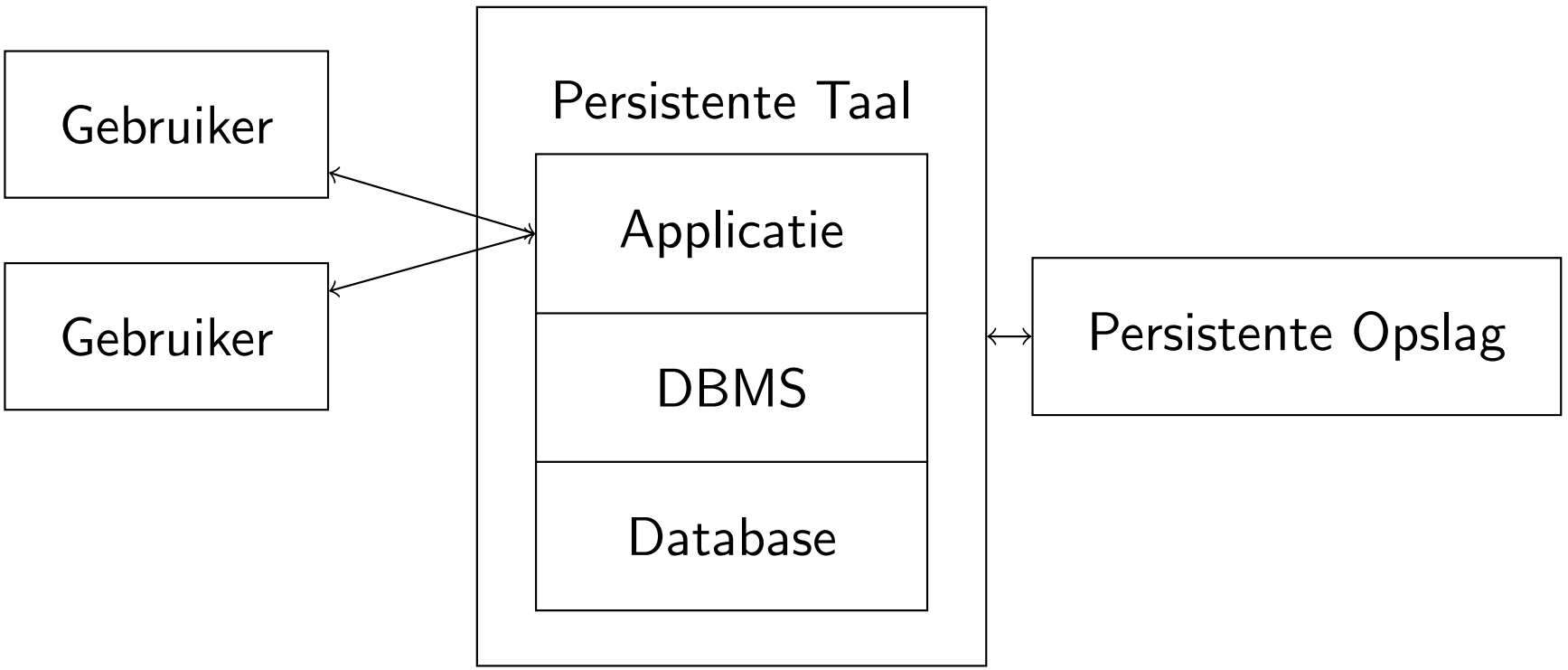
Traditionele Architectuur



Limitaties

- Most DBMS's only support isolation partially.
- Application and DBMS have different type system.
- Serial interface between application and DBMS.
- Distributed system complicates implementation.
- DBMS's are vulnerable to command injection attacks.
- Systeem is moeilijk te verifiëeren.

Mijn Aanpak


Gebruiker
Gebruiker
Persistente Taal
Applicatie
DBMS
Database
Persistente Opslag

Doelen

Onderzoek

Ontwikkeling van:

(1) Een functionele taal voor transactie verwerking.

(2) Methoden om transacties gelijktijdig te verwerken.

(3) Methoden om toestanden efficient op te slaan.

Doelen

Onderzoek

Ontwikkeling van:

(1) Een functionele taal voor transactie verwerking.

(2) Methoden om transacties gelijktijdig te verwerken.

(3) Methoden om toestanden efficient op te slaan.

- Prototype implementatie

Doelen

Onderzoek

Ontwikkeling van:

(1) Een functionele taal voor transactie verwerking.

(2) Methoden om transacties gelijktijdig te verwerken.

(3) Methoden om toestanden efficient op te slaan.

- Prototype implementatie
- Experimenten

Doelen

Onderzoek

Ontwikkeling van:

(1) Een functionele taal voor transactie verwerking.

(2) Methoden om transacties gelijktijdig te verwerken.

(3) Methoden om toestanden efficient op te slaan.

- Prototype implementatie
- Experimenten

Inhoud

Outline

Functioneel Transactieverwerkingsmodel

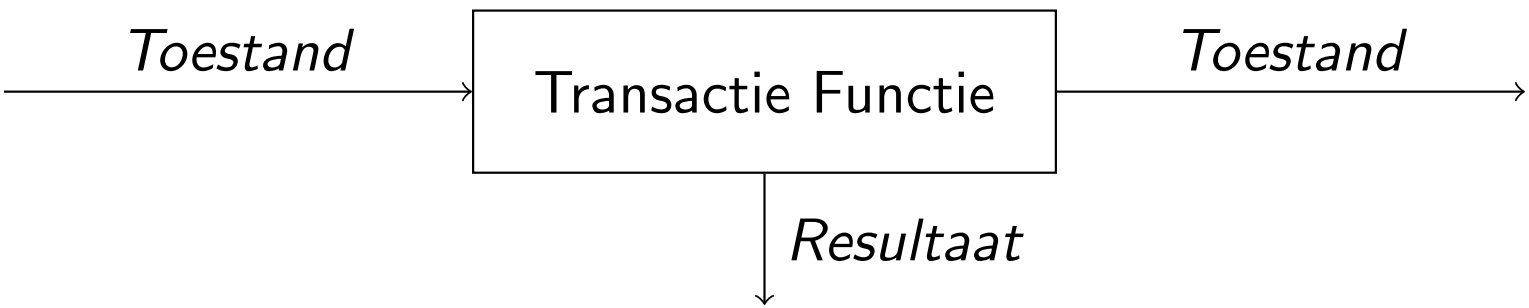
Transactie

Een *transactie functie* is een functie met type:
$Toestand \rightarrow Toestand \times Resultaat$.

Functioneel Transactieverwerkingsmodel

Transactie

Een *transactie functie* is een functie met type:
$Toestand \rightarrow Toestand \times Resultaat$.



Functioneel Transactieverwerkingsmodel

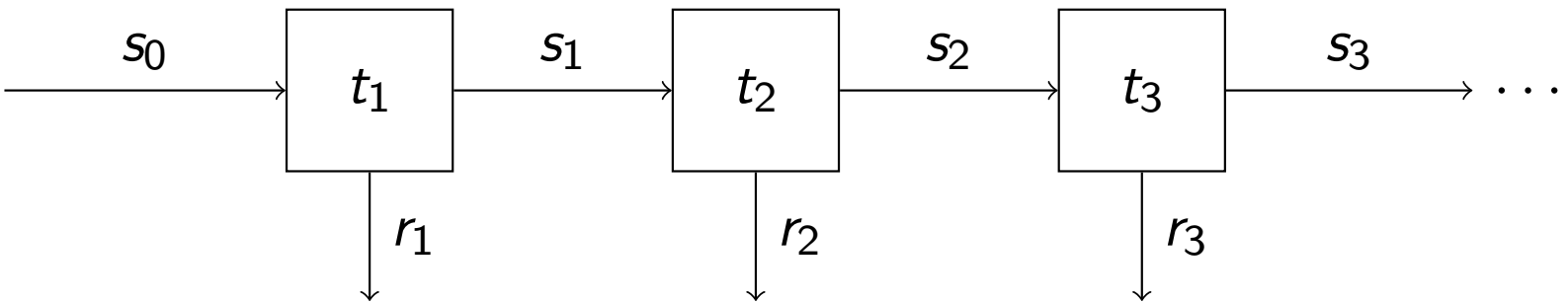
Transactieverwerkingssysteem

Een *transactie manager functie* is een functie met type:
$Toestant \times [Transactie] \rightarrow [Resultaat]$.

Functioneel Transactieverwerkingsmodel

Transactieverwerkingssysteem

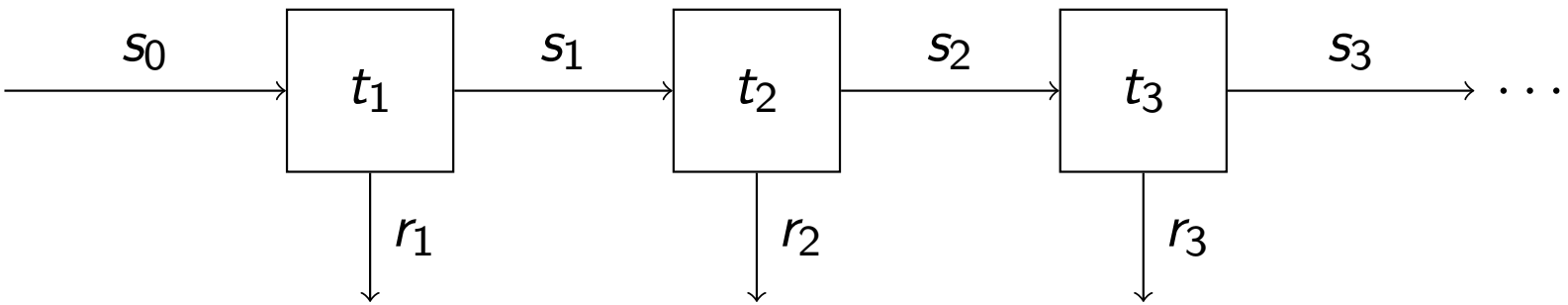
Een *transactie manager functie* is een functie met type:
$Toestant \times [Transactie] \rightarrow [Resultaat]$.



Functioneel Transactieverwerkingsmodel

Transactieverwerkingssysteem

Een *transactie manager functie* is een functie met type:
$Toestant \times [Transactie] \rightarrow [Resultaat]$.



Gelijktijdig uitvoeren van transacties

De toestanden s_i worden *lui* (lazy) geconstrueerd, gedreven door reductie van de resultaten r_i.

Outline

Transactionele Functionele Taal

Toestand

De toestand is een verzameling van bindingen $x = E$, waar:

- x is een *naam*.
- E is een *expressie*.

Transactionele Functionele Taal

Toestand

De toestand is een verzameling van bindingen $x = E$, waar:

- x is een *naam*.
- E is een *expressie*.

Voorbeeld

```
a       = 5
```

Transactionele Functionele Taal

Toestand

De toestand is een verzameling van bindingen $x = E$, waar:

- x is een *naam*.
- E is een *expressie*.

Voorbeeld

```
a       = 5
namen   = ["henk", "piet", "jan"]
```

Transactionele Functionele Taal

Toestand

De toestand is een verzameling van bindingen $x = E$, waar:

- x is een *naam*.
- E is een *expressie*.

Voorbeeld

```
a       = 5
namen   = ["henk", "piet", "jan"]
lengte  = λ lijst .  match lijst
                []  -> 0
                (x:xs) -> 1 + lengte(xs)
```

Transactionele Functionele Taal

Transacties

Een *transactie* is een verzameling van bindingen $x = E$, waar

- x is een variable.
- E is een expressie.

Transactionele Functionele Taal

Transacties

Een *transactie* is een verzameling van bindingen $x = E$, waar

- x is een variable.
- E is een expressie.

Variabelen

- *Huidige toestand variabelen*: `x`, `y`, `z`, ...

Transactionele Functionele Taal

Transacties

Een *transactie* is een verzameling van bindingen $x = E$, waar

- x is een variable.
- E is een expressie.

Variabelen

- *Huidige toestand variabelen*: `x`, `y`, `z`, ...
- *Volgende toestand variabelen*: `x'`, `y'`, `z'`, ...

Transactionele Functionele Taal

Transacties

Een *transactie* is een verzameling van bindingen $x = E$, waar

- x is een variable.
- E is een expressie.

Variabelen

- *Huidige toestand variabelen*: `x`, `y`, `z`, ...
- *Volgende toestand variabelen*: `x'`, `y'`, `z'`, ...
- *Resultaat variabele*: `result`

Transactionele Functionele Taal

Voorbeeld

s_0: `namen = ["henk", "piet"]`

Transactionele Functionele Taal

Voorbeeld

s_0: `namen = ["henk", "piet"]`

t_1: `namen' = "jan" : namen`

`result = namen'`

Transactionele Functionele Taal

Voorbeeld

s_0: `namen = ["henk", "piet"]`

t_1: `namen' = "jan" : namen`

`result = namen'` $\rightarrow$ `["jan", "henk", "piet"]`

Transactionele Functionele Taal

Voorbeeld

s_0: `namen = ["henk", "piet"]`

t_1: `namen' = "jan" : namen`

`result = namen'` $\rightarrow$ `["jan", "henk", "piet"]`

s_1: `namen = ["jan", "henk", "piet"]`

Transactionele Functionele Taal

Voorbeeld

s_0: `namen = ["henk", "piet"]`

t_1: `namen' = "jan" : namen`

`result = namen'` $\rightarrow$ `["jan", "henk", "piet"]`

s_1: `namen = ["jan", "henk", "piet"]`

t_2: `lengte' =` λ `lijst . match lijst`

```
  [] -> 0
  (x:xs) -> 1 + lengte'(xs)
```

Transactionele Functionele Taal

Voorbeeld

s_0: `namen = ["henk", "piet"]`

t_1: `namen' = "jan" : namen`

`result = namen'` $\rightarrow$ `["jan", "henk", "piet"]`

s_1: `namen = ["jan", "henk", "piet"]`

t_2: `lengte' =` λ `lijst . match lijst`

```
  [] -> 0
  (x:xs) -> 1 + lengte'(xs)
```

s_2: `namen = ["jan", "henk", "piet"]`

`lengte =` λ `lijst . match lijst`

```
  [] -> 0
  (x:xs) -> 1 + lengte(xs)
```

Transactionele Functionele Taal

Voorbeeld

s_0: `namen = ["henk", "piet"]`

t_1: `namen' = "jan" : namen`

`result = namen'` $\rightarrow$ `["jan", "henk", "piet"]`

s_1: `namen = ["jan", "henk", "piet"]`

t_2:
```
lengte' = λ lijst . match lijst
  [] -> 0
  (x:xs) -> 1 + lengte'(xs)
```

s_2:
```
namen = ["jan", "henk", "piet"]
lengte = λ lijst . match lijst
  [] -> 0
  (x:xs) -> 1 + lengte(xs)
```

t_3: `result = lengte namen`

Transactionele Functionele Taal

Voorbeeld

s_0: `namen = ["henk", "piet"]`

t_1: `namen' = "jan" : namen`

`result = namen'` $\rightarrow$ `["jan", "henk", "piet"]`

s_1: `namen = ["jan", "henk", "piet"]`

t_2:
```
lengte' = λ lijst . match lijst
  [] -> 0
  (x:xs) -> 1 + lengte'(xs)
```

s_2:
```
namen = ["jan", "henk", "piet"]
lengte = λ lijst . match lijst
  [] -> 0
  (x:xs) -> 1 + lengte(xs)
```

t_3: `result = lengte namen` $\rightarrow 3$

Outline

Implementatie

Luie Evaluatie

Luie evaluatie van $f(2 \times 3)$, waar $f(x) = x + x$:

Implementatie

Luie Evaluatie

Luie evaluatie van $f(2 \times 3)$, waar $f(x) = x + x$:

$$f(2 \times 3)$$

Implementatie

Luie Evaluatie

Luie evaluatie van $f(2 \times 3)$, waar $f(x) = x + x$:

$$f(2 \times 3) \rightarrow 2 \times 3 + 2 \times 3$$

Implementatie

Luie Evaluatie

Luie evaluatie van $f(2 \times 3)$, waar $f(x) = x + x$:

$$f(2 \times 3) \rightarrow 2 \times 3 + 2 \times 3 \rightarrow 6 + 2 \times 3$$

Implementatie

Luie Evaluatie

Luie evaluatie van $f(2 \times 3)$, waar $f(x) = x + x$:

$$f(2 \times 3) \rightarrow 2 \times 3 + 2 \times 3 \rightarrow 6 + 2 \times 3 \rightarrow 6 + 6$$

Implementatie

Luie Evaluatie

Luie evaluatie van $f(2 \times 3)$, waar $f(x) = x + x$:

$$f(2 \times 3) \rightarrow 2 \times 3 + 2 \times 3 \rightarrow 6 + 2 \times 3 \rightarrow 6 + 6 \rightarrow 12$$

Implementatie

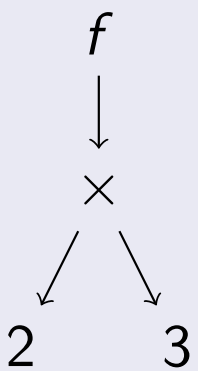
Luie Evaluatie

Luie evaluatie van $f(2 \times 3)$, waar $f(x) = x + x$:

$$f(2 \times 3) \rightarrow 2 \times 3 + 2 \times 3 \rightarrow 6 + 2 \times 3 \rightarrow 6 + 6 \rightarrow 12$$

Graafreductie

Graafreductie van $f(2 \times 3)$:



Implementatie

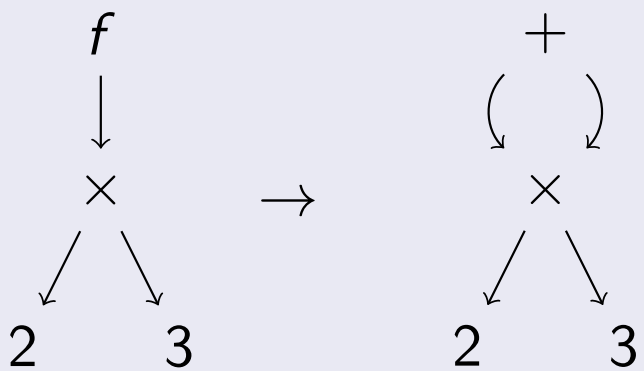
Luie Evaluatie

Luie evaluatie van $f(2 \times 3)$, waar $f(x) = x + x$:

$$f(2 \times 3) \to 2 \times 3 + 2 \times 3 \to 6 + 2 \times 3 \to 6 + 6 \to 12$$

Graafreductie

Graafreductie van $f(2 \times 3)$:



Implementatie

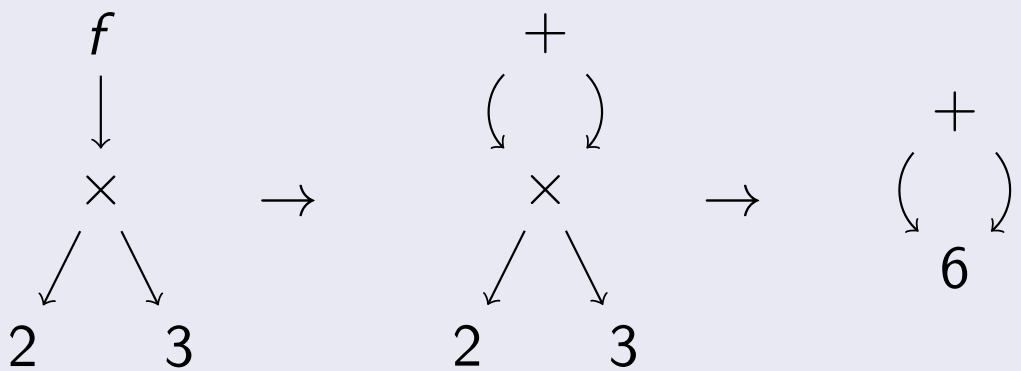
Luie Evaluatie

Luie evaluatie van $f(2 \times 3)$, waar $f(x) = x + x$:

$$f(2 \times 3) \to 2 \times 3 + 2 \times 3 \to 6 + 2 \times 3 \to 6 + 6 \to 12$$

Graafreductie

Graafreductie van $f(2 \times 3)$:



Implementatie

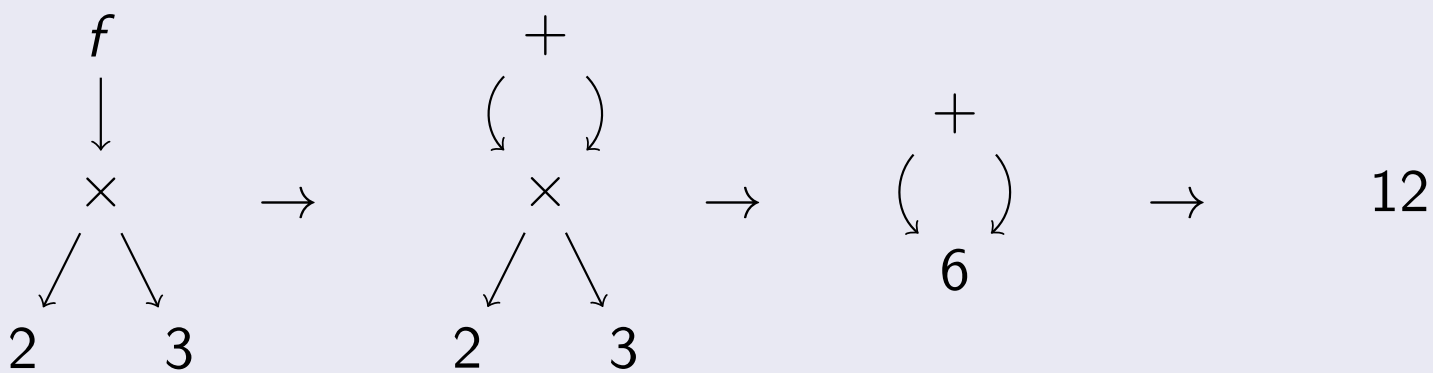
Luie Evaluatie

Luie evaluatie van $f(2 \times 3)$, waar $f(x) = x + x$:

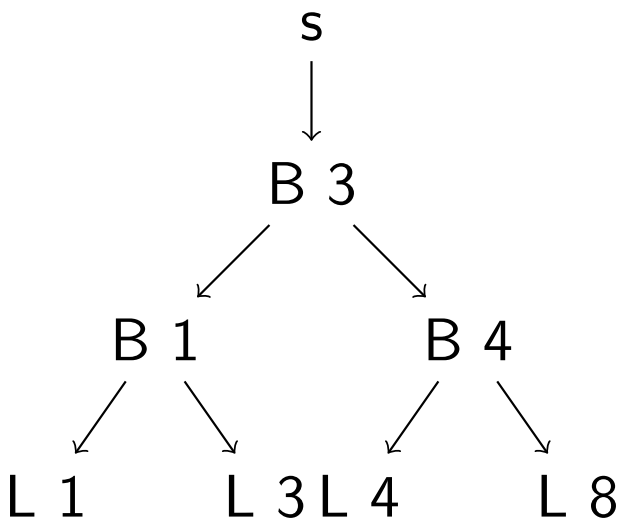
$$f(2 \times 3) \to 2 \times 3 + 2 \times 3 \to 6 + 2 \times 3 \to 6 + 6 \to 12$$

Graafreductie

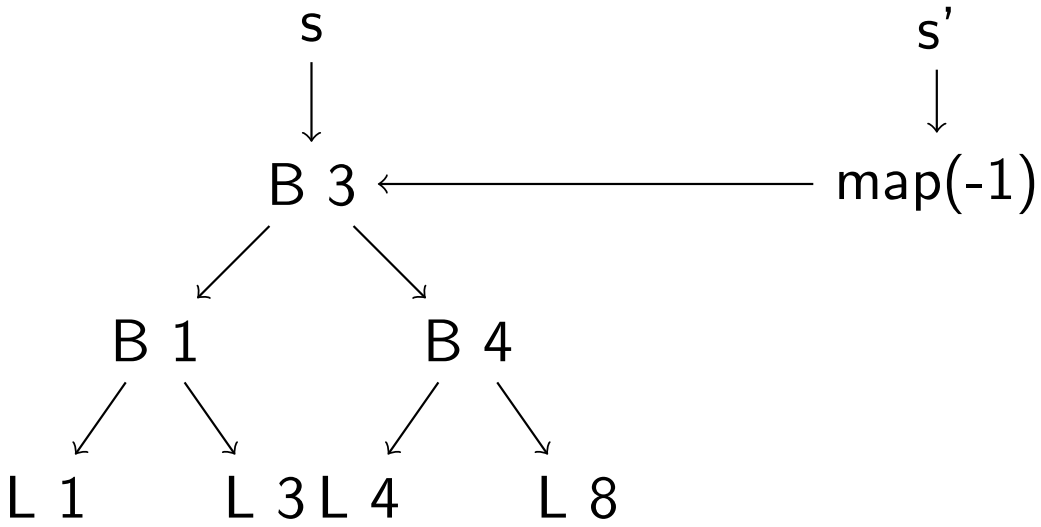
Graafreductie van $f(2 \times 3)$:



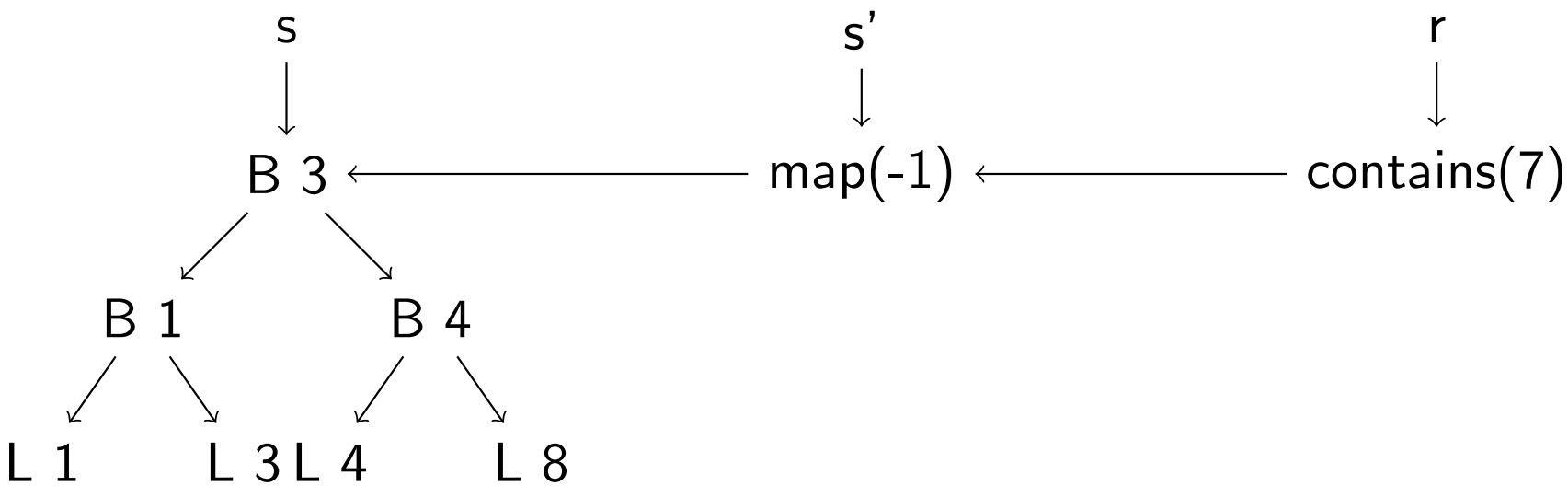
Gelijktijdig Transacties Uitvoeren



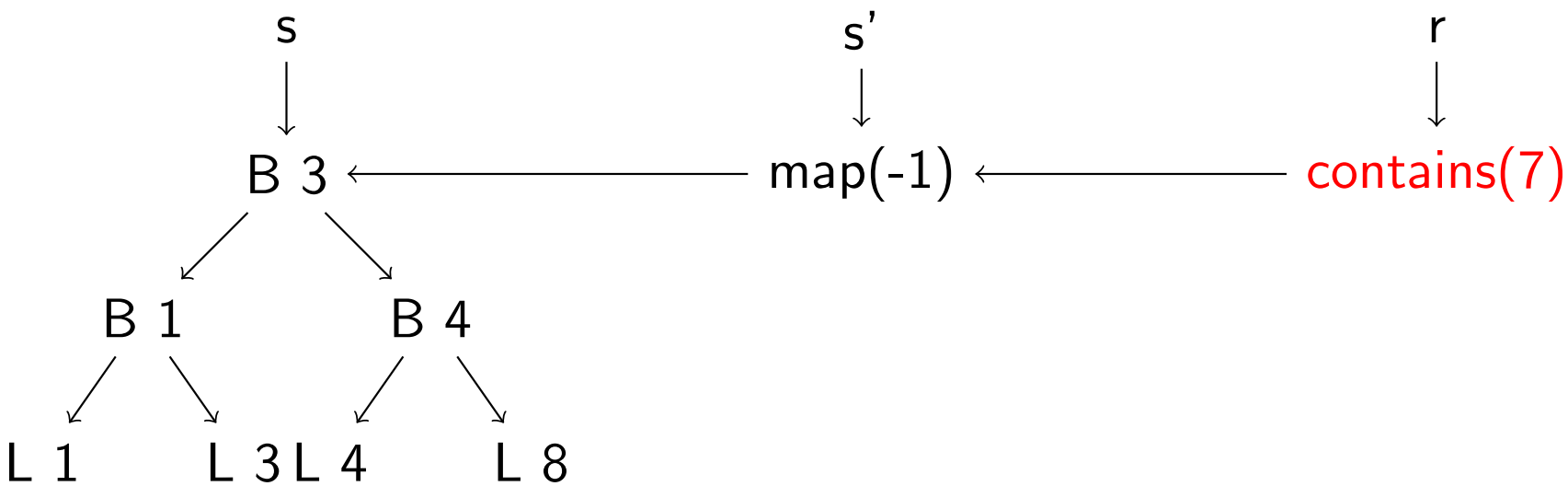
Gelijktijdig Transacties Uitvoeren



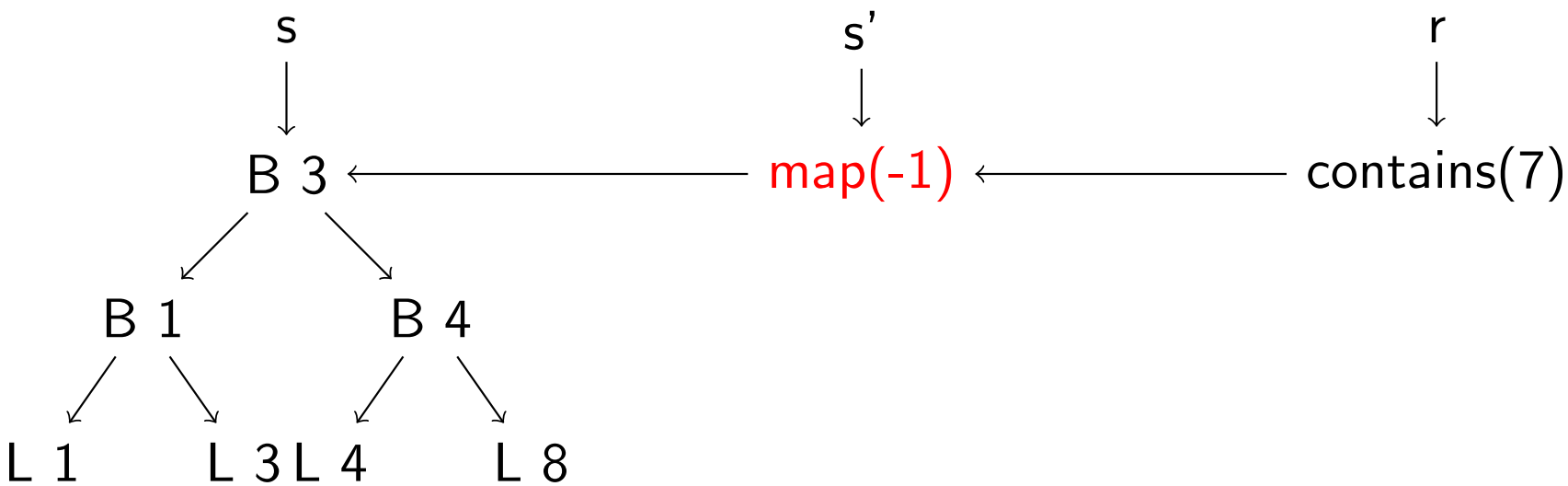
Gelijktijdig Transacties Uitvoeren



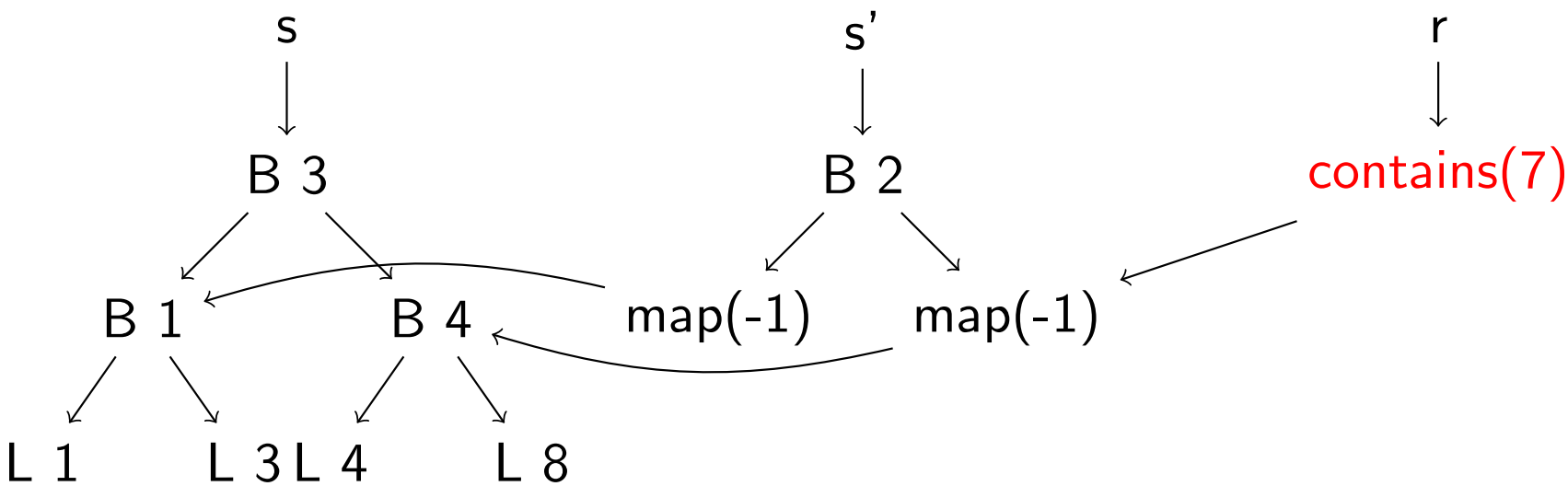
Gelijktijdig Transacties Uitvoeren



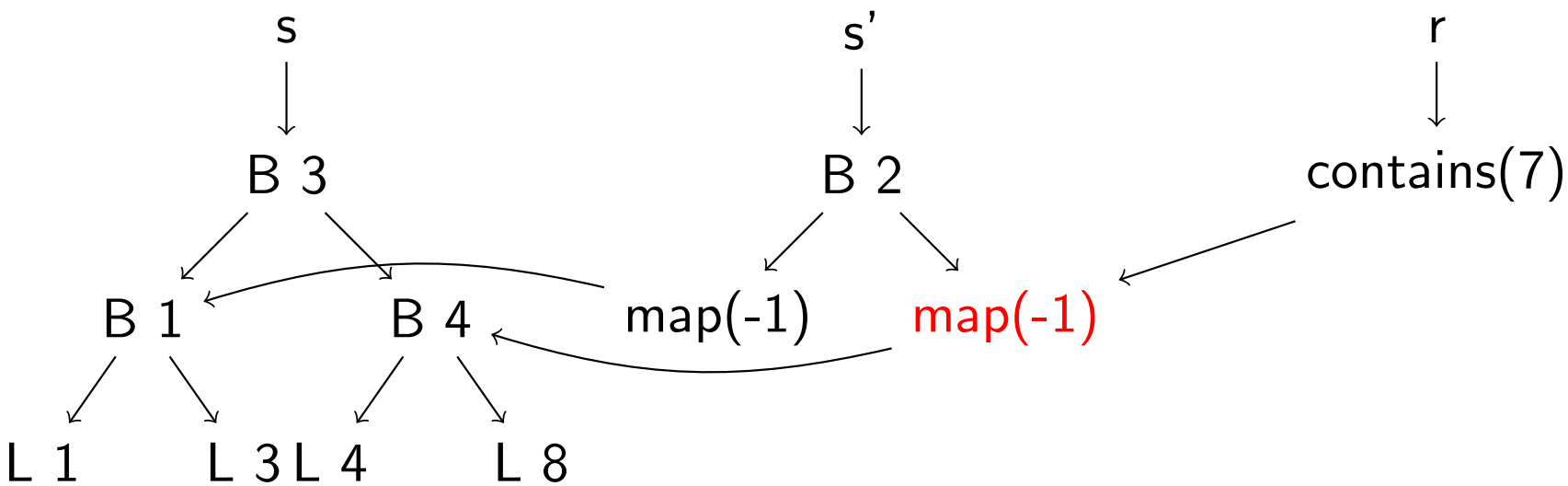
Gelijktijdig Transacties Uitvoeren



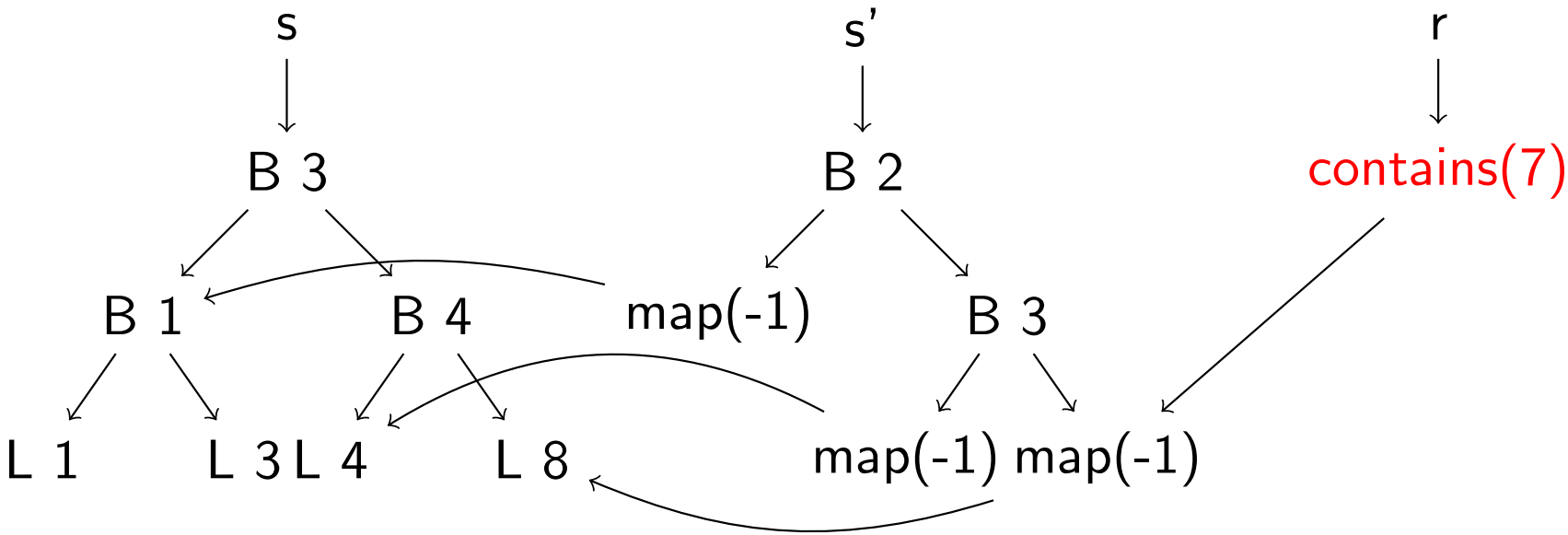
Gelijktijdig Transacties Uitvoeren



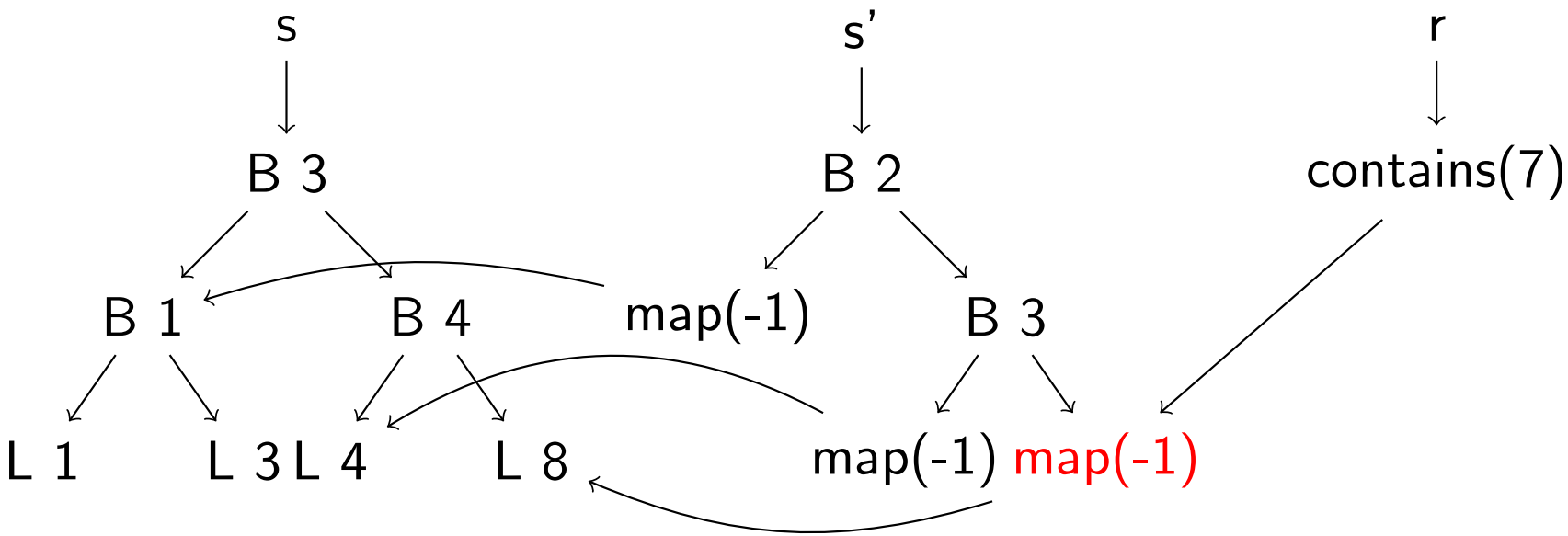
Gelijktijdig Transacties Uitvoeren



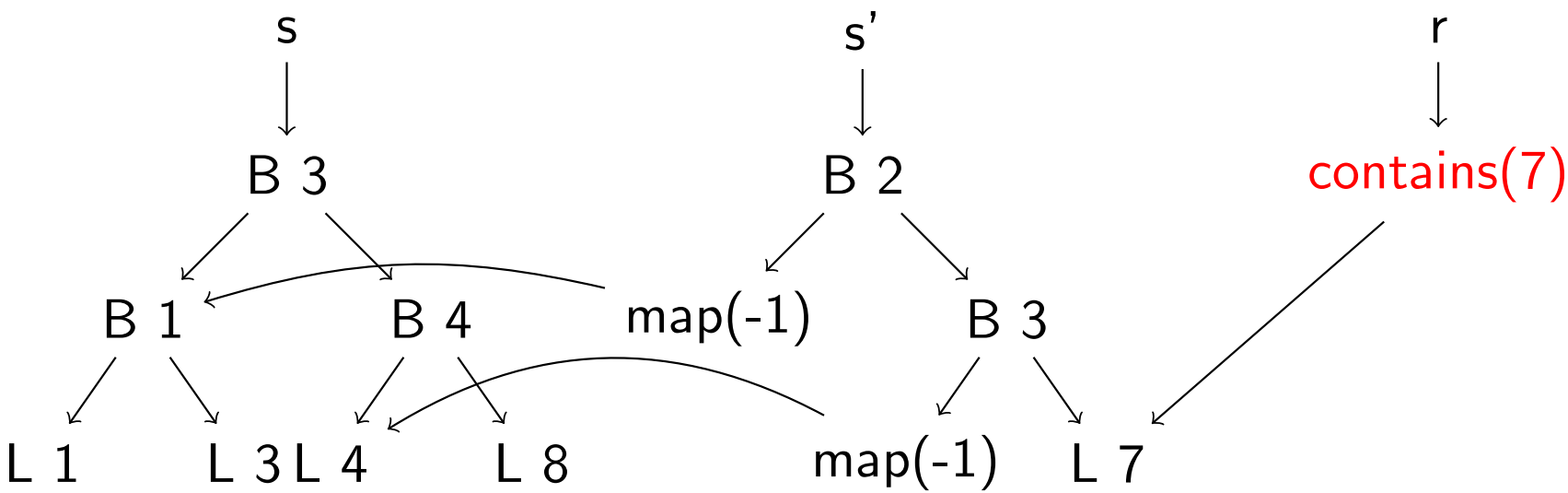
Gelijktijdig Transacties Uitvoeren



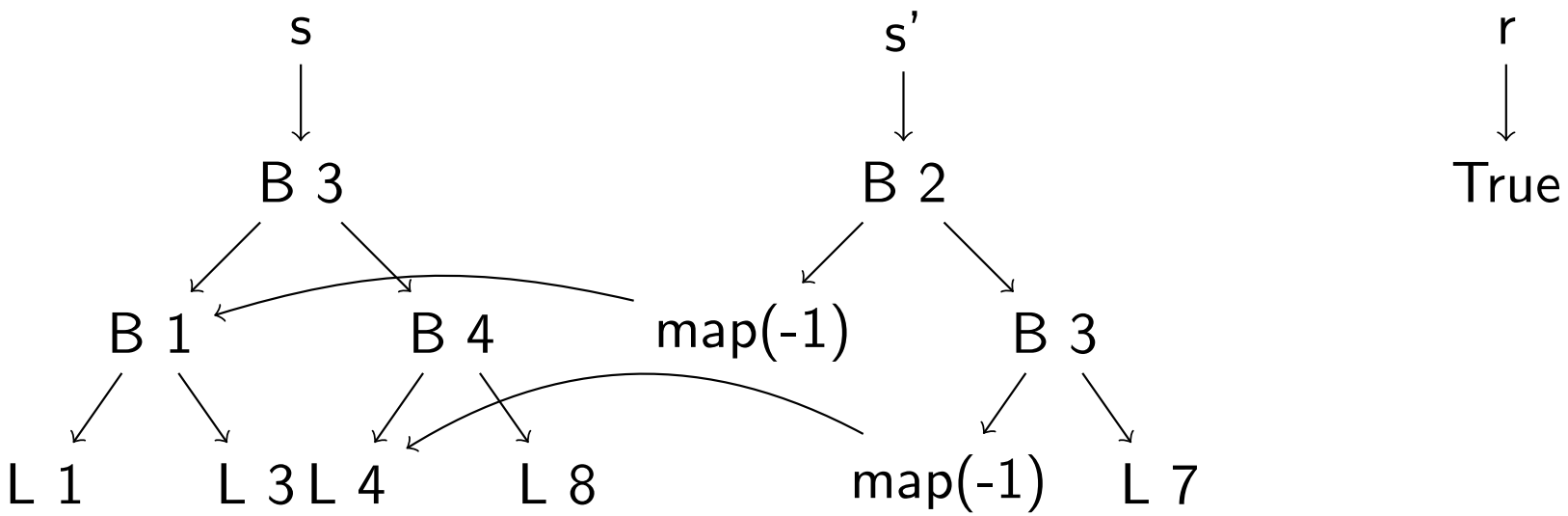
Gelijktijdig Transacties Uitvoeren



Gelijktijdig Transacties Uitvoeren



Gelijktijdig Transacties Uitvoeren



Parallelle Graafreductie

Gelijktijdige en Paralelle Graafreductie

We gebruiken meerdere reductiethreads om:

- Transacties door elkaar uit te voeren zodat ze niet op elkaar hoeven te wachten.
- Transacties tegelijk uit te voeren door meerdere processors te gebruiken.

Parallelle Graafreductie

Taakverdeling

Taakverdeling onder threads

Stricte Functies

Een functie is *strict in een argument* als het de waarde van een argument nodig heeft om toegepast te kunnen worden.

- Voorbeeld: $a + b$ is strict in a en b omdat je beide nodig hebt om de optelling uit te kunnen voeren.

Outline

Maintaining Persistence - Journaling

Journaling

Journaling zorgt voor *atomicity* en *durability*:

- Transacties worden weggeschreven voordat ze uitgevoerd worden.

Maintaining Persistence - Journaling

Journaling

Journaling zorgt voor *atomicity* en *durability*:

- Transacties worden weggeschreven voordat ze uitgevoerd worden.
- Als het systeem crasht, dan kan een toestand *gerecoverd* worden door alle opgeslagen transacties opnieuw uit te voeren.

Maintaining Persistence - Journaling

Journaling

Journaling zorgt voor *atomicity* en *durability*:

- Transacties worden weggeschreven voordat ze uitgevoerd worden.
- Als het systeem crasht, dan kan een toestand *gerecoverd* worden door alle opgeslagen transacties opnieuw uit te voeren.

Checkpointing

Journaling alleen is niet genoeg:

- De log kan erg groot worden

Maintaining Persistence - Journaling

Journaling

Journaling zorgt voor *atomicity* en *durability*:

- Transacties worden weggeschreven voordat ze uitgevoerd worden.
- Als het systeem crasht, dan kan een toestand *gerecoverd* worden door alle opgeslagen transacties opnieuw uit te voeren.

Checkpointing

Journaling alleen is niet genoeg:

- De log kan erg groot worden
- Lange recovery tijden

Maintaining Persistence - Journaling

Journaling

Journaling zorgt voor *atomicity* en *durability*:

- Transacties worden weggeschreven voordat ze uitgevoerd worden.
- Als het systeem crasht, dan kan een toestand *gerecoverd* worden door alle opgeslagen transacties opnieuw uit te voeren.

Checkpointing

Journaling alleen is niet genoeg:

- De log kan erg groot worden
- Lange recovery tijden
- Toestand is gelimiteerd tot werkgeheugen

Maintaining Persistence - Journaling

Journaling

Journaling zorgt voor *atomicity* en *durability*:

- Transacties worden weggeschreven voordat ze uitgevoerd worden.
- Als het systeem crasht, dan kan een toestand *gerecoverd* worden door alle opgeslagen transacties opnieuw uit te voeren.

Checkpointing

Journaling alleen is niet genoeg:

- De log kan erg groot worden
- Lange recovery tijden
- Toestand is gelimiteerd tot werkgeheugen

Oplossing: We maken een checkpoint van een toestand.

Persistence - Snapshotting

Aanpak

Serialiseer de toestand naar een snapshotbestand.

Persistence - Snapshotting

Aanpak

Serialiseer de toestand naar een snapshotbestand.

- Complicatie: Gelijktijdige graafreductie.

Persistence - Snapshotting

Aanpak

Serialiseer de toestand naar een snapshotbestand.

- Complicatie: Gelijktijdige graafreductie.

Voordelen

- We kunnen berekeningen checkpointen.

Persistence - Snapshotting

Aanpak

Serialiseer de toestand naar een snapshotbestand.

- Complicatie: Gelijktijdige graafreductie.

Voordelen

- We kunnen berekeningen checkpointen.
- We kunnen makkelijk sharing behouden.

Persistence - Snapshotting

Aanpak

Serialiseer de toestand naar een snapshotbestand.

- Complicatie: Gelijktijdige graafreductie.

Voordelen

- We kunnen berekeningen checkpointen.
- We kunnen makkelijk sharing behouden.

Nadelen

- Toestand is gelimiteerd tot hoeveelheid werkgeheugen.

Persistence - Snapshotting

Aanpak

Serialiseer de toestand naar een snapshotbestand.

- Complicatie: Gelijktijdige graafreductie.

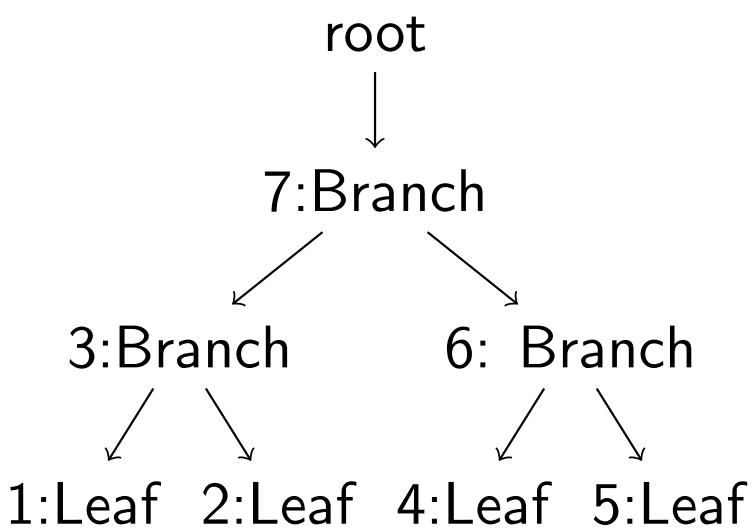
Voordelen

- We kunnen berekeningen checkpointen.
- We kunnen makkelijk sharing behouden.

Nadelen

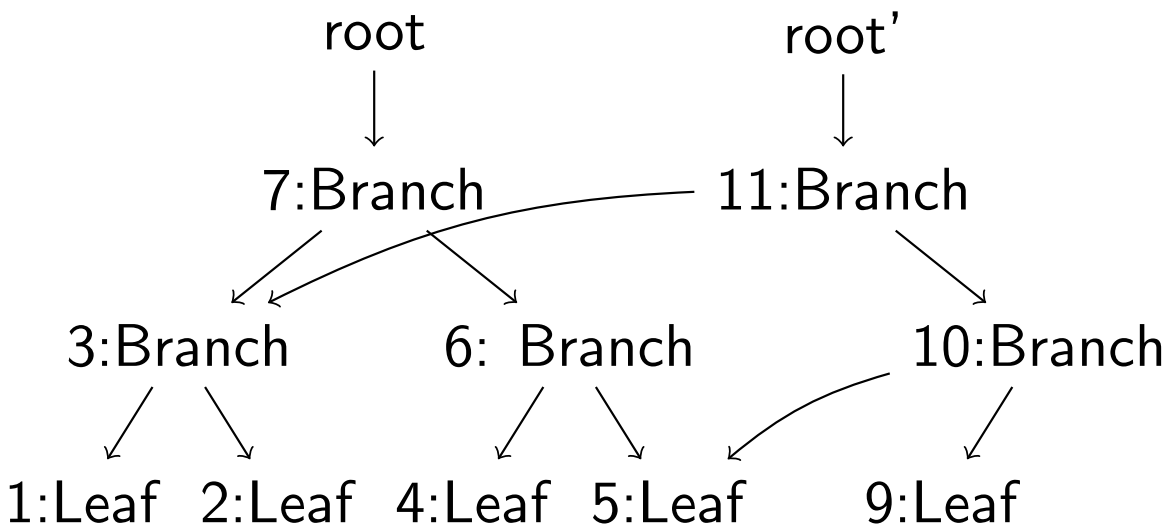
- Toestand is gelimiteerd tot hoeveelheid werkgeheugen.
- Recovery kan lang duren.

Persistence - Log-Structured Storage



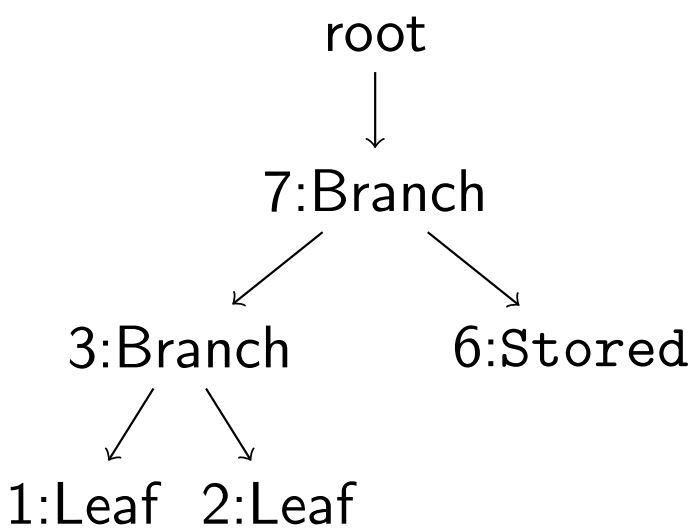

```
1: Leaf
2: Leaf
3: Branch 1 2
4: Leaf
5: Leaf
6: Branch 4 5
7: Branch 3 6
8: Root 7
```

Persistence - Log-Structured Storage




```
1 : Leaf
2 : Leaf
3 : Branch 1 2
4 : Leaf
5 : Leaf
6 : Branch 4 5
7 : Branch 3 6
8 : Root 7
9 : Leaf
10 : Branch 9 5
11 : Branch 3 10
12 : Root 12
```

Persistence - Log-Structured Storage




```
1: Leaf
2: Leaf
3: Branch 1 2
4: Leaf
5: Leaf
6: Branch 4 5
7: Branch 3 6
8: Root 7
```

Persistence - Log-Structured Storage

Voordelen

- Toestanden groter dan werkgeheugen.

Persistence - Log-Structured Storage

Voordelen

- Toestanden groter dan werkgeheugen.
- Snelle recovery.

Persistence - Log-Structured Storage

Voordelen

- Toestanden groter dan werkgeheugen.
- Snelle recovery.

Nadelen / Complicaties

- Checkpointen van berekeningen in duur.

Persistence - Log-Structured Storage

Voordelen

- Toestanden groter dan werkgeheugen.
- Snelle recovery.

Nadelen / Complicaties

- Checkpointen van berekeningen in duur.
- Garbage collection is nodig.

Persistence - Log-Structured Storage

Voordelen

- Toestanden groter dan werkgeheugen.
- Snelle recovery.

Nadelen / Complicaties

- Checkpointen van berekeningen in duur.
- Garbage collection is nodig.
- Sharing behouden tijdens garbage collection is duur.

Persistence - Gecombineerde Aanpak

Aanpak

We delen de heap in twee delen:

Ongereduceerd: Snapshotting

Gereduceerd: Log-structured storage

Persistence - Gecombineerde Aanpak

Aanpak

We delen de heap in twee delen:

Ongereduceerd: Snapshotting

Gereduceerd: Log-structured storage

Voordelen

- We kunnen berekeningen checkpointen.

Persistence - Gecombineerde Aanpak

Aanpak

We delen de heap in twee delen:

Ongereduceerd: Snapshotting

Gereduceerd: Log-structured storage

Voordelen

- We kunnen berekeningen checkpointen.
- Toestanden groter dan werkgeheugen.

Persistence - Gecombineerde Aanpak

Aanpak

We delen de heap in twee delen:

Ongereduceerd: Snapshotting

Gereduceerd: Log-structured storage

Voordelen

- We kunnen berekeningen checkpointen.
- Toestanden groter dan werkgeheugen.
- Snelle recovery.

Persistence - Gecombineerde Aanpak

Aanpak

We delen de heap in twee delen:

Ongereduceerd: Snapshotting

Gereduceerd: Log-structured storage

Voordelen

- We kunnen berekeningen checkpointen.
- Toestanden groter dan werkgeheugen.
- Snelle recovery.

Nadelen

- Garbage collection is nodig.

Persistence - Gecombineerde Aanpak

Aanpak

We delen de heap in twee delen:

Ongereduceerd: Snapshotting

Gereduceerd: Log-structured storage

Voordelen

- We kunnen berekeningen checkpointen.
- Toestanden groter dan werkgeheugen.
- Snelle recovery.

Nadelen

- Garbage collection is nodig.
- Geen sharing in reduced heap.

Outline

Contributies

Transactionele Functionele Taal

- Taal
- Graafreductie die dynamische bindingen ondersteund
- Transactie manager

Contributies

Transactionele Functionele Taal

- Taal
- Graafreductie die dynamische bindingen ondersteund
- Transactie manager

Gelijktijdige Transacties

- Parallele graaf herschrijver
- Concurrent transactie manager

Contributies

Transactionele Functionele Taal

- Taal
- Graafreductie die dynamische bindingen ondersteund
- Transactie manager

Gelijktijdige Transacties

- Parallele graaf herschrijver
- Concurrent transactie manager

Opslag in Persistent Geheugen

- Snapshotting
- Log-structured Storage
- Gecombineerde Aanpak

Vervolgonderzoek

Onderzoekspunten

(1) Implementatie van opslagtechnieken.
(2) Gebruik van het systeem.
(3) Afhandelen van run-time fouten.
(4) Type checking.
(5) Concurrent data structuren.
(6) Optimistic concurrent control.
(7) Geheugen gebruik verminderen.
(8) Parallele graafreductie.
(9) Scheduling van reductie threads.
(10) Gedistribueerde systemen.