Copyright ©1998, 1999 Timothy Howard Merrett
Permission to make digital or hard copies of part or all of
this work for personal or classroom use is granted without fee
provided that copies are not made or distributed for profit or
commercial advantage and that copies bear this notice and
full citation in a prominent place. Copyright for components
of this work owned by others than T. H. Merrett must be
honoured. Abstracting with credit is permitted. To copy
otherwise, to republish, to post on servers, or to redistribute
to lists, requires prior specific permission and/or fee. Re-
quest permission to republish from: T. H. Merrett, School
of Computer Science, McGill University, fax 514 398 3883.

The author gratefully acknowledges support from the tax-
payers of Québec and of Canada who have paid his salary
and research grants while this work was developed at McGill
University, and from his students (who built the implement-
ations and investigated the data structures and algorithms)
and their funding agencies.

T. H. Merrett  ©98/11
Concurrent Relational Programming

The Sequential (--) and Parallel (||) Combinators

\[
\text{relation } \text{CommonRel}(\text{attr}) \leftarrow \{(0)\};
\]

\[
\text{update } \text{CommonRel} \text{ change } \text{attr} \leftarrow \text{attr} + 1 ||
\]

\[
\text{update } \text{CommonRel} \text{ change } \text{attr} \leftarrow \text{attr} - 1;
\]

This executes the two statements in any order ("parallel"), and leaves \text{CommonRel} unchanged because the operations cancel.

\[
\text{relation } \text{CommonRel}(\text{attr}) \leftarrow \{(0)\};
\]

\[
( \text{ LocalRel1 } \leftarrow \text{CommonRel} --
\]

\[
\text{update } \text{LocalRel1} \text{ change } \text{attr} \leftarrow \text{attr} + 1 --
\]

\[
\text{CommonRel } \leftarrow \text{ LocalRel1 }
\]

\) ||

\[
( \text{ LocalRel2 } \leftarrow \text{CommonRel} --
\]

\[
\text{update } \text{LocalRel2} \text{ change } \text{attr} \leftarrow \text{attr} - 1 --
\]

\[
\text{CommonRel } \leftarrow \text{ LocalRel2 }
\]

\);

This has 20 possible interleavings. Two give the right result, 0; 9 give -1; and 9 give 1.
Concurrent Relational Programming

Synchronization

Synchronization is by a blocking T-selector:

if the result of an ordinary T-selector would be empty, the process blocks until the operand is changed to make the result not empty.

The \textbf{where} is replaced by \textbf{when}.

T. H. Merrett
Synchronization

This idea from *Linda* (Carreiro & Gelernter, CACM 32 (1989) 444):

```plaintext
out("a string", 15.01, 17, "another string")
... puts a “tuple” into “tuple space”.
rd("a string", ? f, ? i, "another string")
... reads it, assigning values to the variables;
blocks if no corresponding tuple;
nondeterministically returns one, if many tuples.
in("a string", ? f, ? i, "another string")
... like rd but consumes the tuple.
```

T. H. Merrett

©99/11
Synchronization

\[
\text{relation } tSpace(S1, N1, N2, S2) \leftarrow \\
\{("a string", 15.01, 17, "another string")\}; \\
synch \leftarrow \text{when } S1="a string" \\
\text{ and } S1="another string" \text{ in } tSpace;
\]

N.B. Does not consume; reads all. 
\(\text{(Nondeterminism is from another operator, another syntax, pick).}\)

T. H. Merrett
Semaphores (Dijkstra, CACM 11 (1968) 341)

\[ P(s) \text{ “proberen”} \quad \text{if } s.\text{cnt} = 0 \text{ then WAIT} \]
\[ \text{dec}(s.\text{cnt}) \]

\[ V(s) \text{ “vrijgeven”} \quad \text{inc}(s.\text{cnt}) \]
\[ \text{waken waiters} \]

T. H. Merrett
Semaphores

Here is a semaphore,

\[
\begin{align*}
\text{relation } & \text{SEMAPHORE(Sem\_name, Sem\_count);} \\
\text{proc } & \text{I(sema) is } \{ \text{SEMAPHORE } \lhd \text{ sema; } \}; \\
\text{proc } & \text{P(sema) is } \\
& \{ \text{S } \leftarrow \text{ when Sem\_count } > 0 \text{ in } \\
& \quad (\text{SEMAPHORE } \text{ijoin } [\text{Sem\_name} \text{ in sema});} \\
& \quad \text{update } \text{SEMAPHORE } \text{change Sem\_count } \leftarrow \\
& \quad \quad \text{Sem\_count } - 1 \text{ using } ([\text{Sem\_name} \text{ in sema});} \\
& \}; \\
\text{proc } & \text{V(sema) is } \\
& \{ \text{update } \text{SEMAPHORE } \text{change Sem\_count } \leftarrow \\
& \quad \text{Sem\_count } + 1 \text{ using } ([\text{Sem\_name} \text{ in sema});} \\
& \};
\end{align*}
\]

T. H. Merrett 98/11,99/11
Semaphores

and here we use it as a *mutex*.

```plaintext
relation Sem1(Sem_name, Sem_count)←
   {("sem1", 1)};
I(in Sem1);
relation CommonRel(attr) ← {0};
( P(in Sem1) --
   LocalRel1 ← CommonRel --
   update LocalRel1 change attr← attr+1 --
   CommonRel ← LocalRel1 --
   V(in Sem1)
 ) ||
( P(in Sem1) --
   LocalRel2 ← CommonRel --
   update LocalRel2 change attr← attr−1 --
   CommonRel ← LocalRel2 --
   V(in Sem1)
 );
```

T. H. Merrett
A Brief History of Concurrency Mechanisms

Coroutines
SIMULA 67
Conway, 1963

Semaphores
"THE" multiprogramming system
Dijkstra, 1968
(Conditional) Critical Region
(Conditional) Critical Region
(Conditional) Critical Region
(replaced by Monitors)
Hoare, 1972
Hoare, 1973
Hoare, 1974

Monitors

Guarded Commands
Brinch Hansen, 1972

Path Expressions
Brinch Hansen, 1972
(limited, but cf. ALGOL 68)
Brinch Hansen, 1973
Brinch Hansen, 1973

CSP (Communicating Sequential
Dijkstra, 1975
Processes)
Campbell

DP (Distributed Processes)
& Haberman, 1974
Haberman, 1974

T. H. Merrett

©99/11