Simulating *tccp* in Mozart

Thomas Sjöland, Seif Haridi,
Per Brand, Erik Klintskog

SICS and KTH/Teleinformatics

CLP/CCP systems from SICS

- SICStus Prolog
  - classical LP
  - global constraints
- Mozart–Oz
  - networked programs
  - multiparadigm, oo+cp
  - higher order
  - constraint solvers

How to integrate time

- Reasoning
  - reasoning (with constraints) over time
- Synchronization
  - reactive programming
    - time–outs, delays, watchdogs etc.

Mozart–Oz

- Mozart–Oz
  - CCP + distributed objects
  - explicit parallel threads on many sites
  - higher order programming

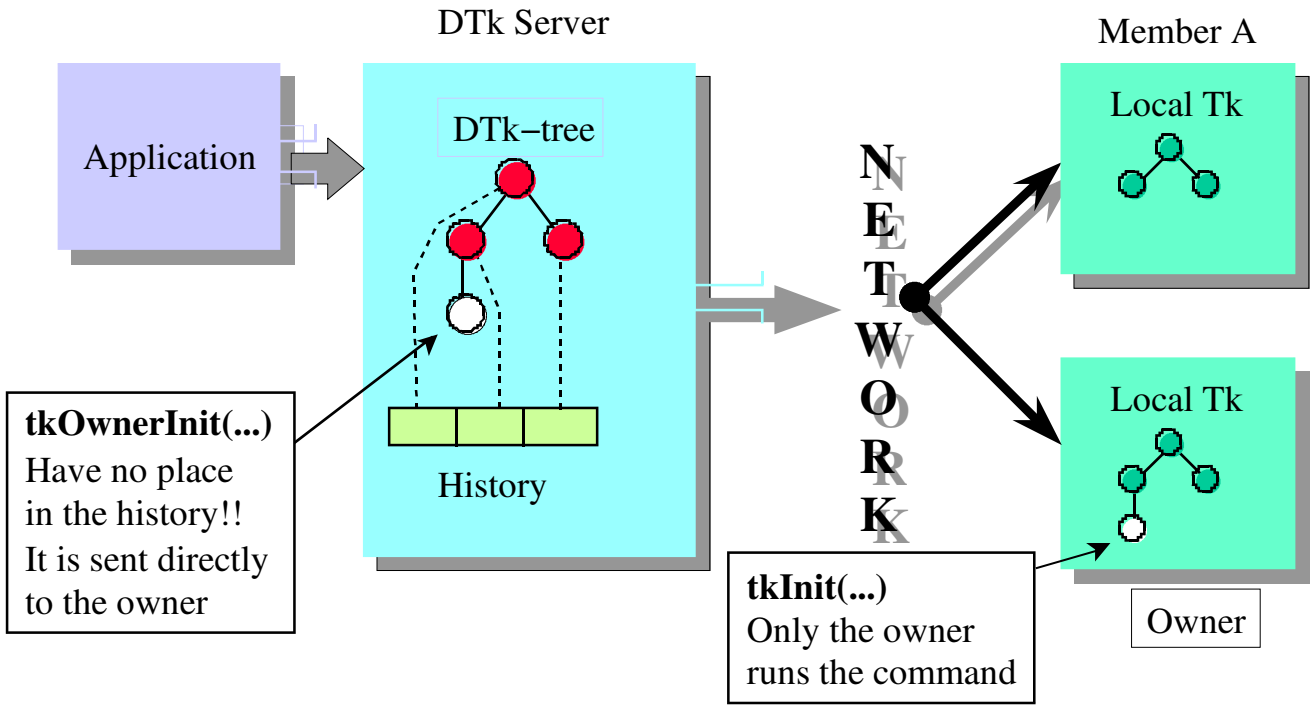
Mozart–Oz – Distribution

- Transparent distribution
- mobile objects (and code)
- distributed unification
- distributed garbage collection
- **Allows tCCP embedding**

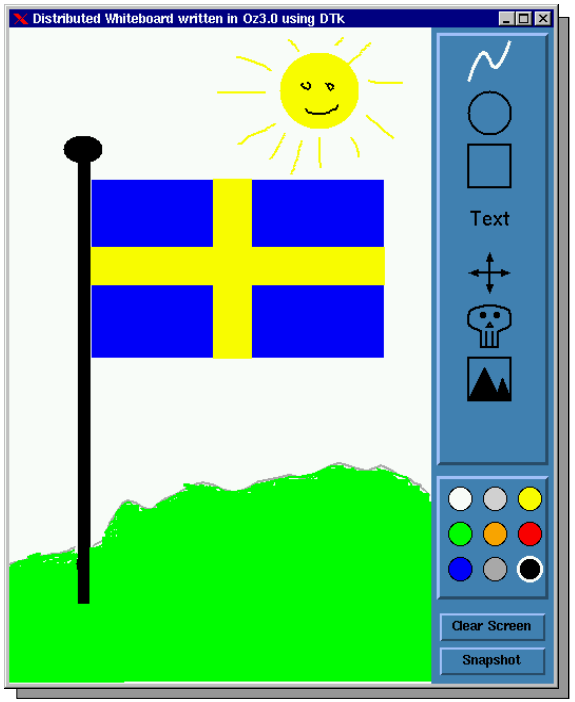
Experimental Mozart–applications

- Tool that transforms a single–user application to an internet–aware multi–user collaborative application
- Internet tools
  - distributed whiteboard for collaboration
  - instant messaging
  - distributed slide presentation
  - chess board

Example of Local Behaviour



Whiteboard



- Members can select their own tool and pencolor. These selections are stored locally at the client side
- They share the global drawing area on which they work on

tCCP embedding in Mozart

- assumptions of tCCP
  - bounded time rewrite steps
    - $\zeta 1 \hookrightarrow \ldots \hookrightarrow \zeta i \hookrightarrow \ldots$
  - totally ordered timed system
    - rewrite $\leftrightarrow$ clock tick
  - maximal parallellism, not interleaving
    - unbounded number of processors
    - zero time distribution of tasks

Discrete events in Mozart

- The simulation package gives
  - synchronisation primitive `{T}` for ”clock tick”
  - In one clock tick
    - Mozart executes until
      - all threads are delayed or suspended
    - Then clock ”ticks” and the process continues

tCCP expressed in Mozart

- Full Mozart computation in each step
- `now c then A else B`
  - primitive implementing all timed ops
    - delays
    - timeout
    - watchdogs

Translation tccp –> Mozart

- ρ(tell(c))
-
  - {T} ρ(c)

Non–blocking entail test

```
{Check proc {$} Constraint end}
```

- **`true`** if **`Constraint`** entailed
- **`false`** otherwise (immediately)

Translation tccp –> Mozart

- $\rho\left(\sum_{i=1}^{n} \text{ask}(c_i) \text{ -> } A_i\right)$
-
- `cond` $\rho(c_1)$ `then {T}` $\rho(A_1)$
- `[]`
- `...`
- `[]` $\rho(c_n)$ `then {T}` $\rho(A_n)$
- `end`

Translation tccp –> Mozart

- ρ(now c then A else B)
-
- `if {Check proc {$} ρ(c) end}`
- `  then ρ(A)`
- `  else ρ(B)`
- `end`

Translation tccp –> Mozart

- ρ(A||B)
-
- `thread` ρ(A) `end` ρ(B)

Translation tccp –> Mozart

- $\rho(\exists^{d} x A)$
-
- `local Y in` $\rho(A)[x/Y]$ `end`

Translation tccp –> Mozart

- ρ(p(x) :– A)
-
- p → `proc {$ ρ(x)} ρ(A) end`

Translation tccp –> Mozart

- ρ(p(x))
- {T} {ρ(p) ρ(x)}

Translation tccp –> Mozart

- ρ (p)
- *M[*p*]*
- replace p by corresponding procedure

Translation tccp –> Mozart

- ρ(x)
- x
- replace x by corresponding varname

Finite analysis of computations

- Given an abstract interpretation $<\alpha, \gamma, \mapsto, \multimap\bullet>$
- $\zeta 1 \mapsto \ldots \mapsto \zeta i \mapsto \ldots \subseteq \gamma(\alpha(\zeta 1 \mapsto \ldots \mapsto \zeta i \mapsto \ldots))$
- and a function o computing observables
- such that
- $\exists m \forall i[\alpha(\zeta 1 \mapsto \ldots \mapsto \zeta i \mapsto \ldots) \subseteq \alpha(\zeta 1) \multimap\bullet \ldots \multimap\bullet \alpha(\zeta m)]$
- and $\exists m \forall i[o(\zeta 1 \mapsto \ldots \mapsto \zeta i \mapsto \ldots) \subseteq o(\gamma(\alpha(\zeta m)))]$
- then *we can finitely abstract all observables*
- *from one computable abstract state*

Conclusion

- tCCP is naturally translated into Mozart
- using simulation primitives
-
- Discrete event semantics