CSP++ : Software Synthesis from Formal Specifications

Bill Gardner, Alicia Gumtie, John Carter
School of Computer Science, University of Guelph

Tool available for free download:
http://www.uoguelph.ca/~gardnerw/csp++/

Background

formal methods:
- using mathematically-based languages and techniques to specify, model, implement, and verify systems
- *Advantage:* code is provably correct
- *Disadvantage:* perceived difficulty and cost; hard to find enough experts

***selective* formalism:**
- a compromise between full-formal and non-formal development
- only apply formal methods where most beneficial: modelling concurrency, common source of problems (e.g., deadlocks)
- SW synthesis eliminates manual translation from formal specs to source code
- minimal gurus required; ordinary C++ developers still useful!

CSP:
- Communicating Sequential Processes (Hoare)
- process-oriented, event-based formal algebra
- process defined as *sequence* of abstract events and/or hierarchical *composition* (parallel or sequential) of other processes
- interprocess communication via message passing on channels (nonbuffered, unidirectional)
- synchronization occurs on events and channel I/O
- verify formal properties (deadlock, livelock, refinement) via third-party tools
- Timed CSP adds operators for delay, interrupt, timeout, suitable for soft real-time systems

CSP++

what it is:
- object-oriented application framework (OOAF) plus translator
- classes implement semantics of CSP primitives—processes, events, and channels
- **cspt** translator synthesizes C++ from CSP spec for compilation with OOAF library
- developers supply C++ user-coded functions (UCFs) to overload events in CSP control backbone
- latest version supports Timed CSP operators
- compatible with *NIX platforms with GNU g++ and Pth (portable threading library)

example:

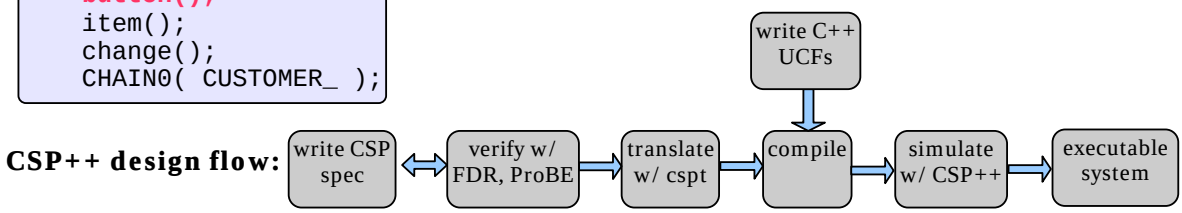
```
CUSTOMER = coin!25 -> button -> item -> change -> CUSTOMER
```

```
void button_ucf(ActionType t, ActionRef* a, Var* v, Lit* l) {
    cout << "The button was pushed!" << endl;
}
```

CSP process + UCF on 'button' event

```
AGENTPROC( CUSTOMER_ )
   coin() << 25;
   button();
   item();
   change();
   CHAIN0( CUSTOMER_ );
```

synthesized C++!



CSP++ design flow:

FDR and ProBE are verification tools from Formal Systems (Europe) Ltd.