CCL in Pictures

James Ivers, Kurt C. Wallnau

Sponsored by the U.S. Department of Defense
© 2005 by Carnegie Mellon University
Example

Starting with a simple case
- single runtime appEnv
- inner is a subassembly
  - but you can’t tell from here
Example – Define Component Type

```ccl
component C() {
    ...
}
```

the shell of a component type
Example – Define Component Type

```
// CCL
component C() {
    sink async toc();
    source unicast tic();
    source unicast talk();
}
```

stimulus received on sink pins and emitted on source pins

*pin signature is covered in a separate module*
Example – Define Component Type

```ccl
component C() {
    sink async toc();
    source unicast tic();
    source unicast talk();

    threaded react passIt(...) {
        ... 
    }
}
```

Component behavior is specified by reaction. Reactions may be units of concurrent execution. Components may have one or more reactions.
Example – Define Component Type

```ccl
// CCL
component C() {
  sink asynch toc();
  source unicast tic();
  source unicast talk();
}
```

```ccl
threaded react passIt(toc, tic, talk) {
  ...}
```

Each pin defines two events:
- \(^{\text{pin start of interaction}}\)
- \(^{\text{pin end of interaction}}\)

Events can be observed:
- on transition triggers
  - \(^{\text{toc}}, \text{tic}, \text{talk}\)

Events can be generated:
- as an action
  - \(^{\text{tic}}, \text{talk}, \text{toc}\)

The behavior of a reaction is visible through its pins.
Example – Define Component Type

```
// CCL
component C() {
  sink asynch toc();
  source unicast tic();
  source unicast talk();

  threaded react passIt(toc, tic, talk) {
    start->ready{}
    ready->work{
      trigger ^toc(); action ^tic();
      work->log{
        trigger $tic(); action ^talk();
        log->ready{
          trigger $talk(); action $toc();
        }
      }
    }
  }
}
```

behavior is specified as executable state machines ≃ UML statecharts
details about the CCL action language and writing reactions are provided in a separate module.
Example – Define Environment Type

\[
\text{environment } E() \{
\}
\]

the shell of an environment type

an virtual palette associated with each environment type
Example – Define Environment Type

// CCL
environment E() {
    service Clock() {} 
    service Log() {}
}

an virtual palette of services types
associated with each environment type
Example – Define Environment Type

```ccl
environment E() {
    // CCL
    service Clock() {
        source unicast tic();
        threaded react ticking(tic) {
            start->run{}
            run->run1{trigger after(10); action ^tic();}
            run1->run{trigger $tic();}
        }
    }
}
```

services are environment-provided component types
services may have different well-formedness rules
Example – Define Environment Type

// CCL
environment E() {

    service Clock() {
        source unicast tic();
        threaded react ticking(tic) {
            start->run{}
            run->run1{trigger after(10); action ^tic();}
            run1->run{trigger $tic();}
        }
    }

    service Log() {
        sink asynch listen();
        threaded react logging(listen) {
            start->run{}
            run->run{trigger ^listen(); action $listen();}
        }
    }
}
Example – Define Environment Type

```
// CCL
function E() {
  environment E() {
    service Clock() {
      source unicast tic();
      threaded react ticking(tic) {
        start->run{}
        run->run1{trigger after(10); action ^tic();}
        run1->run{trigger $tic();}
      }
    }
    service Log() {
      sink async listen();
      threaded react logging(listen) {
        start->run{}
        run->run{trigger ^listen(); action $listen();}
      }
    }
    sink logic {->, {->,} {...} ...}
    source logic {->}
  }
  sink handler (async, mutex) { } //details omitted
}
```

sink handlers is not covered here...

© 2005 by Carnegie Mellon University
Example – Define Inner Assembly Type

```
// CCL
assembly Inner ()(E) {
...
}
```

Each assembly type is parameterized by an environment type. The environment type supplies interaction mechanisms and services.
Example – Define Inner Assembly Type

```ccl
assembly Inner()(E) {
    assume {
        E:Clock innerClock();
        E:Log innerLog();
    }
}
```

assembly types may assume environment provided service instances
Example – Define Inner Assembly Type

```
// CCL
assembly Inner () (E) {
  assume {
    E:Clock innerClock();
    E:Log innerLog();
  }
}
```

assembly types may assume environment provided service instances
Example – Define Inner Assembly Type

```ccl
assembly Inner()(E) {
    assume {
        E:Clock innerClock();
        E:Log innerLog();
    }
    C c1(), c2(); // instantiation
}
```

assemblies also comprise component instances
Example – Define Inner Assembly Type

// CCL
assembly Inner() (E) {  
  assume  
    E:Clock innerClock();
    E:Log innerLog();
  }

  C c1(), c2(); // instantiation
  innerClock:tic ~> c1:toc;
  c1:tic ~> c2:toc;
  c1:talk ~> innerLog:listen;
  c2:talk ~> innerLog:listen;
}

interactions take place between a source pin and one or more sink pins
connections must conform on
- source/sink pin signature
- connector-imposed restrictions
- reasoning framework restrictions

details about interaction and well-formedness checking is covered in a different module.
Example – Define Inner Assembly Type

// CCL
assembly Inner()(E) {
    assume {
        E:Clock innerClock();
        E:Log innerLog();
    }

    C c1(), c2(); // instantiation

    innerClock:tic ~> c1:toc;
    c1:tic ~> c2:toc;
    c1:talk ~> innerLog:listen;
    c2:talk ~> innerLog:listen;

    expose {c2:tic as drip}
}

an assembly type by default has no visible interactive behavior
Example – Define Inner Assembly Type

The resulting assembly type \texttt{Inner} looks like a component
• with the addition of resource assumptions
• assumptions must be discharged at instantiation time
  - by passing along the assumption
  - by satisfying the assumption
Example – Define Outer Assembly Type

```ccl
assembly Outer()(E) {
    assume {
        E:Clock outerClock();
        E:Log outerLog();
    }
}
```

as before...an assembly makes assumptions about its environment
Example – Define Outer Assembly Type

// CCL
assembly Outer ()(E) {
assume {
    E:Clock outerClock();
    E:Log outerLog();
}
}

Inner inner() {
    Inner:innerClock = outerClock;
    Inner:innerLog = outerLog;
}

Assumptions are satisfied at instantiation time
- syntax: assumed resource = provided resource
- assumed resource in the scope of instantiated assembly type
- provided resource in the scope of the instantiator
Example – Define Outer Assembly Type

```
// CCL
assembly Outer (E) {
    assume {
        E:Clock outerClock();
        E:Log outerLog();
    }

    Inner inner()
    { 
        Inner:innerClock = outerClock;
        Inner:innerLog = outerLog;
    }

    C c3();
}
```

Components and assemblies can engage in *apparent* peer interaction.
Example – Define Outer Assembly Type

// CCL
assembly Outer ()(E) {
  assume {
    E:Clock outerClock();
    E:Log outerLog();
  }

  Inner inner() {
    Inner:innerClock = outerClock;
    Inner:innerLog = outerLog;
  }

  C c3();

  inner:drip ~> c3:toc;
  c3:talk ~> outerLog:listen;
}
Example – Define Outer Assembly Type

// CCL
assembly Outer ()(E) {
    assume {
        E:Clock outerClock();
        E:Log outerLog();
    }
    Inner inner() {
        Inner:innerClock = outerClock;
        Inner:innerLog = outerLog;
    }
}

C c3();
inner:drip ~> c3:toc;
c3:talk ~> outerLog:listen;

this outermost assembly has no visible interactive behavior

© 2005 by Carnegie Mellon University
Example – Instantiate Application

```ccl
E appEnv() {
    ...
}
```

instantiate the environment
Example – Instantiate Application

// CCL
E appEnv() {
    E:Clock appClock();
    E:Log appLog();
};

elaboration clause instantiates the services of appEnv
Example – Instantiate Application

// CCL
E appEnv() {
    E:Clock appClock();
    E:Log appLog();
}

Outer myApp() (...) {...}

instantiate the assembly
Example – Instantiate Application

```ccl
// CCL
E appEnv() {
    E:Clock appClock();
    E:Log appLog();
}
};

Outer myApp()(appEnv) {...}
```

deploy the assembly (instance) in an environment (instance)
Example – Instantiate Application

```ccl
// CCL
E appEnv() {
    E:Clock appClock();
    E:Log appLog();
}

Outer myApp()(appEnv) {
    Outer:outerClock = appEnv:appClock;
    Outer:outerLog = appEnv:appLog;
}
```

at elaboration satisfy the assumptions of the assembly type
Example Specification

assembly Inner ()(E) {
    assume {
        E:Clock innerClock();
        E:Log innerLog();
    }
    C c1(), c2(); // instantiation

    innerClock:tic ~> c1:toc;
    c1:tic ~> c2:toc;
    c1:talk ~> innerLog:listen;
    c2:talk ~> innerLog:listen;

    exposure {c2:tic as drip}
}

assembly Outer ()(E) {
    assume {
        E:Clock outerClock();
        E:Log outerLog();
    }
    Inner inner()

    Inner:innerClock = outerClock;
    Inner:innerLog = outerLog;
};
C c3();

inner:drip ~> c3:toc;
C c3();
c3:talk ~> outerLog:listen;

exposure {}
}

E appEnv() { E:Clock appClock(); E:Log appLog(); }

Outer myApp() (appEnv) {
    Outer:outerClock = appEnv:appClock;
    Outer:outerLog = appEnv:appLog; }

© 2005 by Carnegie Mellon University