Introduction to LUSTRE and LUKE

22c181 Spring 2008
Background

- Developed in 1980’s at Verimag (Fr)
- Currently used by Estrel Technologies in Scade development tools
  - Airbus, nuclear power plants
Synchronous Dataflow

Lustre is a synchronous dataflow language

- Synchronous – outputs are “instantaneous”
  - Designed to quickly react to environment
  - "Realtime” or “reactive”
  - Ex: Esterel, Statecharts
- Dataflow – changes force propagation
  - Ex: Spreadsheets

- Simple, modular, functional
Language

- **Nodes**
  - Programs or subprograms
  - Collections of flow definitions

- **Flows (or streams)**
  - infinite sequence of values
  - Defined equationally (no cycles)
node name (parameters) returns (return_vals);

[var local_variable_list;]

let

    flow definition;

    flow definition;

...

tel
Basic Types

- **bool**
  - and, or, not, xor

- **int**

- **real**
  - +, -, *, /, div, mod, =, <=>, <, <=, >, >=, int, real

- **Tuples**
  - Arbitrary combinations of bool, int, real, & tuple terms
  - Used to return multiple values
Luke Tool

- Command line simulator & verifier
  - Fragment of lustre(v4) language
    - does not support arrays, const, assert, #, when, current, real
    - allows nonstandard structures
      - nodes with no inputs
      - =, <> can be used on type bool
  - Outputs simulations & counterexamples to javascript webpage
Luke Usage

- Simulation:
  \texttt{luke --node top_node filename}

- Verification:
  \texttt{luke --node top_node --verify filename}

  - returns either “Valid. All checks succeeded. Maximal depth was $n$” or “Falsified output ‘X’ in node ‘Y’ at depth $n$” along with a counterexample.
Other Operators

- **pre** (previous)
  - *pre X ::* previous value of X

- **->** (followed by)
  - *X -> Y ::* value of X in first step, then the value of Y
  - Generally used together:
    - *X = Y -> pre Z*

- **if ... then ... else**

- **-- ::** single-line comment
Clocks

- Used to delay sampling, execution
- current, when
  - $X = \text{current} \ (Y \ \text{when} \ B)$
  - is not always equivalent to $X = \text{if} \ B \ \text{then} \ Y \ \text{else} \ \text{pre} \ X$
- Not supported by Luke

<table>
<thead>
<tr>
<th></th>
<th>Y</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$X = Y$ when B</td>
<td></td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>current X</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>
Odds & Ends

(Not supported by Luke)

- **assert**(X);
  - When verifying, this statement restricts flows to abide by the Boolean statement X
    - Put known limits on input values

- **const**
  - declare (global) constants

- # :: at most one element of a tuple is true

- External functions
SCADE Operators

- SCADE operators (not in version 4):
  - case :: switching
  - fby(x,n,i) :: initialize with i, delay x by n steps
    - Guarded delay
    - i -> pre (i -> ... pre (x))
  - z = conduct(b, n, x, i)
    - Guarded clock change
    - z = i -> if b then current n(x when b) else pre z
Arrays, Recursion

- Supported as syntactic sugar
- See “A Tutorial of Lustre” for more information
Synchronous Observers

- Another program which observes the behavior of the base code
- “Tester node”
- Contains code to determine if properties are true or not
Traffic Light Example

- Pedestrian crossing with a traffic light
- RGY light, walk/don’t walk sign
- Behavior should be...
Traffic Light Properties

- Cars & pedestrians not allowed at the same time
- Only one light color at a time
- Only walk or don’t walk at a time
- Y must come between R & G
- Others...?
Edge Example

- Compare two implementations of FallingEdge node