Control Data Flow Graphs
- Enabling Vertices

Sue Tyerman

School of Computer Science
University of Adelaide
Inputs/outputs/data

- The vertex is where choices are made.
- Edges store and transfer data
  - Both *data* and *control*
  - It is all data
- What things can influence choice?
  - Which inputs?
  - What data on the inputs?
  - What transformations by the vertex?
  - What outputs?
What things can influence choice?

- **Input edge**
  - Connectors: copy, branch, merge, join - no choice
  - Priority: one edge has priority over others
- **Data**
  - Data affects choice of edge: inspection of values on the edge
- **Transformation**
  - Vertex computation determines behaviour
What things can influence choice?

- Output edge
  - Connectors - no choice
  - Priority: one edge has priority over others
  - Extreme example is the *hot potato*
    - get rid of the result ASAP
    - so any edge will do
Some new work (1)

\[ \forall e \in I(v) : \begin{cases} e = [a_1, a_2...a_n] & \text{if } R(e) \\ e = \bot & \text{if } \neg R(e) \end{cases} \]

\[ \text{Head}\{e\} = \begin{cases} (a_1) & \text{if } R(e) \\ \bot & \text{if } \neg R(e) \end{cases} \]

\[ \text{Head}(X) = (\text{Head}(e_1), \text{Head}(e_2)...\text{Head}(e_n)) \text{ where} \]

\[ X \subseteq I(v) \]
Input function

- Input function on a vertex returns sets of sets of edges
- These sets of edges are possible enabling sets
- This is a dynamic activity, with the composition of the set of sets altering as input edges acquire data
- The guard belonging to each $I$ can examine the readability of edges as well as the values on the edges to determine if a particular enabling is valid.
- All input edges of a vertex are candidates for membership of $X$, not just members of the enabling set.

\[ I(v, \text{Head}(X)) \subseteq \mathcal{P}(\bullet v) = \{\{\}\} \text{ where } v \in V, X \subseteq \bullet v \]

\[ \forall e \in \bullet v : \exists S \in I(v) : e \in S \]
Example

Figure 1: A Simple CDFG showing a vertex with a single input edge and single output edge
Figure 2: A Simple CDFG showing a vertex with three input edges and a single output edge
Figure 3: A Simple CDFG showing a vertex with three input edges and three output edges
Figure 4: A Simple CDFG showing a vertex with a single input edge and three output edges
Power sets of inputs and outputs

- \( I(v_2) = \{ \{ e_1 \} \} \)
- \( O(v_2) = \{ \{ e_2 \} \} \)
- \( I(v_7) = \{ \{ e_4 \} \{ e_5 \} \{ e_6 \} \{ e_4, e_5 \} \{ e_4, e_6 \} \{ e_5, e_6 \} \{ e_4, e_5, e_6 \} \} \)
- \( O(v_7) = \{ \{ e_7 \} \} \)
Power sets of inputs and outputs

- \( I(v_{12}) = \{ \{e_9\},\{e_{10}\},\{e_{11}\},\{e_9, e_{10}\},\{e_9, e_{11}\},\{e_{10}, e_{11}\},\{e_9, e_{10}, e_{11}\} \} \)
- \( O(v_{12}) = \{ \{e_{12}\},\{e_{14}\},\{e_{15}\},\{e_{12}, e_{14}\},\{e_{12}, e_{15}\},\{e_{14}, e_{15}\},\{e_{12}, e_{14}, e_{15}\} \} \)
- \( I(v_{17}) = \{ \{e_{16}\} \} \)
- \( O(v_{17}) = \{ \{e_{18}\},\{e_{19}\},\{e_{20}\},\{e_{18}, e_{19}\},\{e_{18}, e_{20}\},\{e_{19}, e_{20}\},\{e_{18}, e_{19}, e_{20}\} \} \)
Data flow nodes

- Nodes that do not have input or output choice
- Guard evaluates to true if all input edges have values

- \( I(v_2) = \{\{e_1\}\}, \ O(v_2) = \{\{e_2\}\} \)
- \( I(v_7) = \{\{e_4, e_5, e_6\}\}, \ O(v_7) = \{\{e_7\}\} \)
- \( I(v_{12}) = \{\{e_9, e_{10}, e_{11}\}\}, \ O(v_{12}) = \{\{e_{12}, e_{14}, e_{15}\}\} \)
- \( I(v_{17}) = \{\{e_{16}\}\}, \ O(v_{17}) = \{\{e_{18}, e_{19}, e_{20}\}\} \)
End of slide show

Definitions follow
Definitions (1)

A CDFG is a tuple $C = (V, E, I, O)$ where
(V, E) is a connected, directed graph,
$I$ is a function mapping each vertex to an (input) enabling function,
$O$ is a function mapping each vertex to a possible set of outputs.
Definitions (2)

\[ V = \{v_1, \ldots, v_n\} \]

is a finite set whose elements are nodes

\[ E \subseteq V \times V \text{ is an irreflexive flow relation, } \]

whose elements are directed edges

\[ V \cap E = \varnothing \]
Definitions (3)

\[ w = \{(v, w) \in E \mid v \in V, w \in V\} \]

\[ \text{in-deg}(w) = |\bullet w| \]

\[ w\bullet = \{(w, v) \in E \mid v \in V, w \in V\} \]

\[ \text{out-deg}(w) = |w\bullet| \]
Definitions (4)

\[ I(v) \subseteq \mathcal{P}(\bullet v) - \{} \{} \text{ where } v \in V \]
\[ \forall e \in \bullet v : \exists S \in I(v) : e \in S \]

\[ O(v) \subseteq \mathcal{P}(v\bullet) - \{} \{} \text{ where } v \in V \]
\[ \forall e \in v\bullet : \exists T \in O(v) : e \in T \]

\[ \forall e \in E : R(e) \in Boolean, W(e) \in Boolean \]
\[ \forall v \in V : F(v) \in Boolean \]

\[ P = (R, W, F) \]
Definitions (5)

\[ I(v) = \{ \bullet v \} \text{ where } v \in V \]

\[ O(v) = \{ v \bullet \} \text{ where } v \in V \]

For any node \( v \) that is a pure data flow node

\[ |I(v)| = 1 \]

\[ |O(v)| = 1 \]

For any node \( v \) that is not a pure data flow node

\[ |I(v)| > 1 \lor |O(v)| > 1 \]
Definitions (6)

An edge may hold multiple values and so we define

*Size*(e) and *Capacity*(e). Then

\[ R(e) = \text{Size}(e) > 0 \text{ and} \]
\[ W(e) = \text{Size}(e) < \text{Capacity}(e) \]

*If Capacity*(e) is 1, then \( R(e) = \neg W(e) \)

\[ S \in I(v) : \forall e \in S : R(e) \]
\[ en(V) = \]
\[ \{v \in V \mid \neg F(v) \land \exists S \in I(v) : \forall e \in S : R(e)\} \]
\[ T \in O(v) : \forall e \in T : W(e) \]
A node $v$ which is enabled on the basis of $S \in I(v)$ can start firing which causes a change of state from $(R, W, F)$ to $(R', W', F')$

where

$$F'(w) = F(w) \quad \text{if } w \neq v$$
$$F'(v) = true \quad \text{if } w = v$$

and

$$R'(e) = R(e), W'(e) = W(e) \quad \text{if } e \not\in S$$
$$\text{Size}'(e) = \text{Size}(e) - 1,$$
$$W'(e) = \text{Size}'(e) < \text{Capacity}(e),$$
$$R'(e) = \text{Size}'(e) > 0 \quad \text{if } e \in S$$
A node $v$ may complete firing on the basis of $T \in O(v)$ which causes a change of state from $(R, W, F)$ to $(R', W', F')$

where

$$F'(w) = F(w) \quad \text{if } w \neq v$$

$$F'(v) = \text{false} \quad \text{if } w = v$$

and

$$R'(e) = R(e), \quad W'(e) = W(e) \quad \text{if } e \not\in T$$

$$\begin{align*}
\text{Size}'(e) &= \text{Size}(e) + 1, \\
W'(e) &= \text{Size}'(e) < \text{Capacity}(e), \\
R'(e) &= \text{Size}'(e) > 0
\end{align*} \quad \text{if } e \in T$$
Definitions (9)

\[ \forall e \in E : Type(e) \in \Delta, \]

*where \( \Delta \) is an infinite set of types*

\[ \text{queue}(e) = [a_1, a_2 \ldots a_n] \]

\[ \text{Size}(e) = n \text{ where } n \leq \text{Capacity}(e) \]
Definitions (10)

if \( \text{Size}(e) > 0 \) then \( \text{Read}(e) \) transforms \( \text{queue}(e) \) to \( \text{queue}'(e) \)

where \( \text{queue}'(e) = [a_2...a_n] \)

if \( \text{Size}(e) < \text{Capacity}(e) \) then \( \text{Write}(e,x) \) transforms \( \text{queue}(e) \)

to \( \text{queue}'(e) \) where \( \text{queue}'(e) = [a_1, a_2...a_n, x] \)
When a vertex \((v)\) begins firing, the values on the edges in \(S\) are read and initialise the value on the vertex \((\text{Val}(v))\) where

\[
\text{Val}(v) = \text{init}_S(\text{Read}_S(e_1), \text{Read}_S(e_2)\ldots\text{Read}_S(e_n))
\]

where \(S = \{e_1, e_2, \ldots e_n\}\)
Definitions (12)

When a vertex \((v)\) completes firing, the initial value of the vertex \((\text{Val}(v))\) is transformed by the operation \(X\) to \(\text{Val}'(v)\) and this is used to produce the outputs on the edges of \(T\)

\[
\text{Val}'(v) = ((\text{Write}_T(e_1, \text{Val}'(v))), \text{Write}_T(e_2, \text{Val}'(v))...\text{Write}_T(e_n, \text{Val}'(v)))
\]

where \(T = \{e_1, e_2, ...e_n\}\)
Definitions (13)

\[ O(v, \text{Val}(v)) \subseteq \mathcal{P}(v\bullet) - \{\{}\} \text{ where } v \in V \]

\[ | O(v, \text{Val'}(v)) | > 0 \]