An effect type system for modular distribution of dataflow programs

Gwenaël Delaval\textsuperscript{1}   Alain Girault\textsuperscript{1}   Marc Pouzet\textsuperscript{2}

\textsuperscript{1}INRIA Rhône-Alpes, Pop-Art project

\textsuperscript{2}LRI, Demons team

28/11/2006 — Synchron’06
Motivations

Providing a language-oriented solution for the design of functionally distributed systems
Motivations

- Providing a language-oriented solution for the design of functionally distributed systems
  
  Alternative: separate design of each computing resource.
  
  Problems arised:
  
  - One function can involves several computing sites
    → separate design of closely related components, risks of unconsistencies of data.
Motivations

- Providing a language-oriented solution for the design of functionally distributed systems

Alternative: separate design of each computing resource.

Problems arisen:

- One function can involve several computing sites ⇒ separate design of closely related components, risks of inconsistencies of data.
- One computing resource can be involved in several functions ⇒ duplicated control jeopardizing the modularity.
Motivations

Providing a language-oriented solution for the design of functionally distributed systems

Alternative: separate design of each computing resource.
Problems arised:

- One function can involves several computing sites \(\Rightarrow\) separate design of closely related components, risks of inconsistencies of data.
- One computing resource can be involved in several functions \(\Rightarrow\) duplicated control jeopardizing the modularity.

Performing modular distribution:

- to avoid inlining everything
- to allow, by mean of high-order features, dynamic reconfiguration of a site by another (by sending functions through channels)
Example
Multichannel reception system of a software-defined radio

- FPGA
  - filter_1800
- DSP
  - dem_GMSK
- GPP
  - CRC/conv

Reception channel composed of three components:
- a pass-band filter implemented on a FPGA;
- a demodulator implemented on a DSP;
- further processing (e.g., error correction) on a general-purpose processor.
Principles

- Allowing the expression of the localization of computations:
  \texttt{do \ y = f(x) \ at \ P;}

- Inference of the localization of each value and computation from expressed ones: “coloration” of the program. (P. Caspi, A. Girault and D. Pilaud, 1999)
Distribution of synchronous dataflows programs

Principles

- Allowing the expression of the localization of computations:
  \[ \text{do } y = f(x) \text{ at } P; \]
- Inference of the localization of each value and computation from expressed ones: “coloration” of the program. (P. Caspi, A. Girault and D. Pilaud, 1999)

Realisation

- Localization of a value $\leftrightarrow$ spatial type of this value
- “coloration” $\leftrightarrow$ type inference
Spatial types: overview

Type and effect system (J.-P. Talpin and P. Jouvelot, 1992):

- Atomic values: the spatial type of a stream is the set of sites where this stream is present;
- Functional values: spatial types of the form $s_i \rightarrow \langle S, T \rangle \rightarrow s_o$ where
  - $s_i$ is the spatial type of the function’s inputs;
  - $s_o$ is the spatial type of the function’s outputs;
  - $S$ is the set of sites involved in the computation of the function (used for type inference: comprises every site of $s_i$, $s_o$ and $T$);
  - $T$ is a set of communication channels involved in the computation (used for distribution).
Example of spatial types

let node f(x) = y where
  do y1 = f1(x) at P;
  do y2 = f2(y1) at Q;
  do y = f3(y2) at P

Assuming that f1, f2 and f3 are of spatial types
∀α.\{α\} −⟨\{α\}, ∅⟩→ \{α\}, within f:

- f1 and f3 get the spatial type \{P\} −⟨\{P\}, ∅⟩→ \{P\};
- f2 gets the spatial type \{Q\} −⟨\{Q\}, ∅⟩→ \{Q\}
- because of the presence of the two communications, f is finally of spatial type \{P\} −⟨\{P, Q\}, [c_1 : P ↦ Q, c_2 : Q ↦ P]⟩→ \{P\}
let node multichannel_sdr (x,s) = y where
do p = gsm_or_umts(s) at GPP;
match p with
| Gsm ->
do f = filter_bp_1800(x) at FPGA;
do d = demod_GMSK(f) at DSP;
do y = crc_conv(d) at GPP;
| Umts ->
do f = filter_bp_2000(x) at FPGA;
do d = demod_QPSK(f) at DSP;
do c = crc_conv(d) ; y = turbo_code(c) at GPP;
Architecture description

Architecture description: composed of site declarations as symbolic names, with encapsulation and links between sites.

site FPGA;
site DSP;
site GPP;
link FPGA to DSP;
link DSP to GPP;
Architecture description

Architecture description: composed of site declarations as symbolic names, with encapsulation and links between sites.

site FPGA;
site DSP;
site GPP;
link FPGA to DSP;
link DSP to GPP;

The architecture description defines two relations:

- a hierarchy relation $\prec_R: L_1 \prec_R L_2$ iff $L_1$ is a subsite of $L_2$
- a communication relation noted $\mapsto_C: L_1 \mapsto_C L_2$ iff there exists a communication link from $L_1$ to $L_2$. 
Spatial type expressions

\[ H ::= [x_1 : \rho_1, \ldots, x_n : \rho_n] \]

\[ \rho ::= \forall \alpha_1, \ldots, \alpha_n.s \]

\[ s ::= S \mid s \leftarrow \langle S, T \rangle \rightarrow s \mid s \times s \]

\[ S ::= \{\ell_1, \ldots, \ell_n\} \]

\[ \ell ::= L \mid \alpha \]

\[ T ::= [c_1 : L_1 \mapsto L'_1, \ldots, c_n : L_n \mapsto L'_n] \]

Judgments of the form \( H \vdash e : s/S, T \): in the environment \( H \), \( e \) is of “spatial type” \( s \), and computing the expression \( e \)

- involves the set of sites \( S \);
- involves the set of named communication channels \( T \).
Subtyping defines an order on spatial types ($\prec$) : an expression of type $s$ can be used wherever a more general type $s'$ is expected.

$$H \vdash e : s/S, T \quad T' \vdash s \prec s'$$

$$H \vdash e : s'/S, T.T'$$
Subtyping defines an order on spatial types ($\prec$) : an expression of type $s$ can be used wherever a more general type $s'$ is expected.

$$H \vdash e : s/S, T \quad T' \vdash s \prec s'$$

$$H \vdash e : s'/S, T.T'$$

Example: a value of spatial type $S$ can be used as a value of any spatial type $S'$, subset of $S$.

$$S' \subset S$$

$$\emptyset \vdash S \prec S'$$
Subtyping is used to handle typing of communicated values: if the set of communication links $T$ contains a link from $L_1$ to $L_2$, then any value of spatial type $\{L_1\}$ can be used as a value of spatial type $\{L_1, L_2\}$.

\[
\frac{L_1 \leftrightarrow c \rightarrow L_2}{c : L_1 \leftrightarrow L_2 \vdash S \cup \{L_1\} \prec S \cup \{L_1, L_2\}}
\]
A do $D$ at $L$ declaration is typed by constraining the computation of the declaration $D$ to involve at most the site $L$ (or any of its subsites).

$$H \vdash D : H'/S, \forall L' \in S, L' \prec_R L$$

$$H \vdash \text{do } D \text{ at } L : H'/S, T$$
A match $e$ with ... declaration will be duplicated on every site involved in the declarations $D_i$: the expression $e$ must be present on all these sites.

\[
H \vdash e : S/S', \quad T \quad H \vdash D_i : H_i/S_i, \quad T_i \quad S = \bigcup_{i=1}^{n} S_i
\]

\[
\text{match } e \text{ with } \\
H \vdash C_1 \rightarrow D_1 \\
\ldots \\
C_n \rightarrow D_n
\]

\[
\text{match } e \text{ with } \\
H \vdash \begin{array}{l}
C_1 \rightarrow D_1 \\
\ldots \\
C_n \rightarrow D_n
\end{array} : H_1 \uplus \ldots \uplus H_n/S \cup S', \quad T.(T_1 \ldots T_n)
\]
Typing rules: node definitions and applications

Applications: channels are renamed so as to allow multiple instantiations.

\[
H \vdash e_1 : s_1 \leftarrow (S, T) \rightarrow s_2/S_1, T_1 \\
H \vdash e_2 : s_1/S_2, T_2 \\
T' = T[c'_1/c_1, \ldots, c'_n/c_n]
\]

\[
H \vdash e_1(e_2) : s_2/S_1 \cup S_2 \cup S, T_1.T_2.T'
\]

\[
H, x : s \vdash e : s'/S, T
\]

\[
H \vdash \text{let node } f(x) = e : [\text{gen}_H(s \leftarrow (S, T) \rightarrow s')/f]/\emptyset, \emptyset
\]

G. Delaval, A. Girault, M. Pouzet
An effect type system for modular distribution of dataflow programs
Definition of an operation of *projection*:

\[ H \vdash D : H'/S, \ T \xrightarrow{L} D' \]

The declaration \( D \), projected on the site \( L \), results in a new declaration \( D' \).
Definition of an operation of *projection*:

\[ H \vdash D : H'/S, T \xrightarrow{L} D' \]

The declaration \( D \), projected on the site \( L \), results in a new declaration \( D' \)

Communication channels used between two projected declarations are *added as inputs or outputs streams*
Projection of expressions (subtyping rules)

“Forgetting” a site results in the same expression:

\[
\frac{S' \subset S}{\emptyset \vdash e : S \prec S' \Rightarrow e, \emptyset}
\]
Projection of expressions (subtyping rules)

“Forgetting” a site results in the same expression:

$$S' \subset S$$

$$\emptyset \vdash e : S \prec S' \xrightarrow{L} e, \emptyset$$

Using a channel $c$ for communication between two sites results in the definition and use of a stream named as $c$.

$$L_1 \leftrightarrow_c L_2$$

$$c : L_1 \leftrightarrow L_2 \vdash e : S \cup \{L_1\} \prec S \cup \{L_1, L_2\} \xrightarrow{L_1} c, c = e$$

$$L_1 \leftrightarrow_c L_2$$

$$c : L_1 \leftrightarrow L_2 \vdash e : S \cup \{L_1\} \prec S \cup \{L_1, L_2\} \xrightarrow{L_2} c$$
A match/with structure is duplicated on every site where needed:

\[
\begin{align*}
H \vdash e : S/S', T \xrightarrow{L} e', D & \quad L \in S \\
H \vdash D_i : H_i/S_i, T_i \xrightarrow{L} D'_i & \quad S = \bigcup_{i=1}^{n} S_i
\end{align*}
\]

\[
H \vdash \text{match } e \text{ with } \\
\begin{array}{c}
C_1 \rightarrow D_1 \\
\vdots \\
C_n \rightarrow D_n
\end{array}
: H_1 \uplus \ldots \uplus H_n/S \uplus S', T.T_1 \ldots T_n
\]

\[
\xrightarrow{L} D; \\
\begin{array}{c}
\text{match } e' \text{ with } \\
C_1 \rightarrow D'_1 \\
\vdots \\
C_n \rightarrow D'_n
\end{array}
\]
A match/with structure is duplicated on every site where needed:

\[
egin{align*}
H \vdash e : S/S', T \xrightarrow{L} e', D & \quad L \notin S \\
H \vdash D_i : H_i/S_i, T_i \xrightarrow{L} D'_i & \quad S = \bigcup_{i=1}^{n} S_i
\end{align*}
\]

H \vdash D; \\
\begin{array}{l}
\text{match } e \text{ with} \\
C_1 \rightarrow D_1 \\
\vdots \\
C_n \rightarrow D_n
\end{array} : H_1 \cup \ldots \cup H_n/S \cup S', T \cdot T_1 \ldots T_n

\xrightarrow{L} \emptyset
Modular distribution: principle

Channel names, defined or used as streams in the body of a function, will be added to the signature of this function.
Channel names, defined or used as streams in the body of a function, will be added to the signature of this function. This allows multiple instantiations of functions.
Distribution example: node distribution

site A;
site B;
let node f(x) = z where
  do y = x + 1 at A;
  do z = y + 2 at B
Distribution example: node distribution

site A;
site B;
let node f(x) = z where
  do y = x + 1 at A;
  do z = y + 2 at B

spatial type of f: \{A\} \leftarrow \langle\{A, B\}, [c : A \rightarrow B]\rangle \rightarrow \{B\}
Distribution example: node distribution

site A;
site B;
let node f(x) = z where
  do y = x + 1 at A;
  do z = y + 2 at B

spatial type of f: \{A\} \rightarrow \{A, B\}, [c : A \rightarrow B] \rightarrow \{B\}

→ On A, c is added as an output:

let node f(x) = (d, c) where
  y = x + 1;
  c = y
Distribution example: node distribution

site A;
site B;
let node $f(x) = z$ where
  do $y = x + 1$ at A;
  do $z = y + 2$ at B

spatial type of $f$: $\{A\} \rightarrow\langle\{A, B\}, [c : A \mapsto B]\rangle \rightarrow \{B\}$

$\rightarrow$ On $B$, $c$ is added as an input:

let node $f(x, c) = z$ where
  $z = c + 2$
Distribution example: application

\[
\text{let node } g \ (x_1, x_2) = (y_1, y_2) \text{ where} \\
y_1 = f(x_1) ; \\
y_2 = f(x_2)
\]

spatial type of \( g \):

\[
\{A\} \times \{A\} \rightarrow \langle \{A, B\}, \{ c_1 : A \rightarrow B, c_2 : A \leftarrow B \} \rangle \rightarrow \{B\} \times \{B\}
\]
let node g (x1,x2) = (y1,y2) where
  y1 = f(x1);
  y2 = f(x2)

spatial type of g:
\{A\} \times \{A\} \rightarrow \langle\{A, B\}, \{c_1 : A \mapsto B, c_2 : A \mapsto B\} \rangle \rightarrow \{B\} \times \{B\}

⇒ use of *two distinct channels*: one for each application
Application: projection on $A$ and $B$

- On $A$:
  - Channels $c_1$ and $c_2$ as outputs of $f$;
  - $c_1$ and $c_2$ added as outputs of $g$

```
let node g (x1,x2) = (d,d,c1,c2) where
d,c1 = f(x1);
d,c2 = f(x2)
```

- On $B$:
  - Channels $c_1$ and $c_2$ as inputs of $f$;
  - $c_1$ and $c_2$ added as inputs of $g$
Application: projection on A and B

- On A:
  - Channels \( c_1 \) and \( c_2 \) as outputs of \( f \);
  - \( c_1 \) and \( c_2 \) added as outputs of \( g \)

  \[
  \text{let node } g \ (x_1, x_2) = (d, d, c_1, c_2) \text{ where}
  \]

  \[
  \begin{align*}
  d, c_1 &= f(x_1); \\
  d, c_2 &= f(x_2)
  \end{align*}
  \]

- On B:
  - Channels \( c_1 \) and \( c_2 \) as inputs of \( f \);
  - \( c_1 \) and \( c_2 \) added as inputs of \( g \)

  \[
  \text{let node } g \ (x_1, x_2, c_1, c_2) = (y_1, y_2) \text{ where}
  \]

  \[
  \begin{align*}
  y_1 &= f(x_1, c_1); \\
  y_2 &= f(x_2, c_2)
  \end{align*}
  \]
Addition of outputs and inputs channels as output and input of the node:

\[
\begin{align*}
H \vdash e_1 : s_1 & \Rightarrow \langle S, T \rangle \to s_2/S_1, T_1 \xrightarrow{L} e'_1, D_1 \\
H \vdash e_2 : s_1/S_2, T_2 & \xrightarrow{L} e'_2, D_2 \\
T' &= T[c'_1/c_1, \ldots, c'_n/c_n] \\
[c'_1, \ldots, c'_n] &= [c \text{ appearing as channels whose source is } L] \\
[c_1, \ldots, c_m] &= [c \text{ appearing as channels whose target is } L]
\end{align*}
\]

\[
\begin{align*}
H \vdash e_1(e_2) : s_2/S_1 \cup S_2 \cup S, T_1 \cdot T_2 \cdot T' & \xrightarrow{L} x, D_1 ; D_2 ; (x, c'_1, \ldots, c'_n) = e'_1(e'_2, c'_1, \ldots, c'_n)
\end{align*}
\]
Node declaration

Addition of inputs and outputs channels in the signature:

\[ H, x : s \vdash e : s'/S, \quad T \xrightarrow{L} e', \quad D \]

\[ [c_1^o, \ldots, c_m^o] = [c \text{ appearing as channels whose source is } L] \]

\[ [c_1^i, \ldots, c_p^i] = [c \text{ appearing as channels whose target is } L] \]

\[ H \vdash \text{let node } f(x) \text{ : } [\text{gen}_H(s \leftarrow \langle S, T \rangle \rightarrow s')/f]/\emptyset, \emptyset \]

\[ L \xrightarrow{L} \text{let node } f(x, c_1^i, \ldots, c_p^i) \]

\[ = \text{let } D \text{ in } (e, c_1^o, \ldots, c_m^o) \]
Conclusion

Contribution:

- Proposal of a method for modular distribution of dataflow programs, expressed in a language extended with distribution primitives.
Conclusion

- **Contribution:**
  - Proposal of a method for modular distribution of dataflow programs, expressed in a language extended with distribution primitives.

- **Future work:**
  - How to handle polymorphism? (i.e., functions being computed anywhere): simple when such functions involves only one site, but with several sites?
  - Finer architecture description: typed channels,...
  - Examination of higher-order dataflow programs distribution.
  - Goal: expression of dynamic reconfiguration of a site by sending functions through channels.
Thanks !...
Warning ! Appendix...
Typing rules (expressions)

\[
\begin{align*}
H \vdash i : s / \text{sites}(s), \emptyset \\
H \vdash x : s / \text{sites}(s), \emptyset \\
\frac{s \leq H(x)}{H \vdash x : s / \text{sites}(s), \emptyset}
\end{align*}
\]

\[
\begin{align*}
H \vdash e_1 : s_1 / S_1, T_1 \\
H \vdash e_2 : s_2 / S_2, T_2 \\
\frac{H \vdash e_1, e_2 : s_1 \times s_2 / S_1 \cup S_2, T_1 \cdot T_2}{H \vdash e_1, e_2 : s_1 \times s_2 / S_1 \cup S_2, T_1 \cdot T_2}
\end{align*}
\]

\[
\begin{align*}
H \vdash e_1 : s_1 \rightarrow^\langle S, T\rangle s_2 / S_1, T_1 \\
H \vdash e_2 : s_1 / S_2, T_2 \\
T' = T[c'_1/c_1, \ldots, c'_n/c_n] \\
\frac{H \vdash e_1(e_2) : s_2 / S_1 \cup S_2 \cup S, T_1 \cdot T_2 \cdot T'}{H \vdash e_1(e_2) : s_2 / S_1 \cup S_2 \cup S, T_1 \cdot T_2 \cdot T'}
\end{align*}
\]

\[
\begin{align*}
H \vdash D : H'/S_1, T_1 \\
H \vdash D : H'/S_1, T_1 \\
H \vdash e : s / S_2, T_2 \\
\frac{H \vdash \text{let } D \text{ in } e : s / S_1 \cup S_2, T_1 \cdot T_2}{H \vdash \text{let } D \text{ in } e : s / S_1 \cup S_2, T_1 \cdot T_2}
\end{align*}
\]

G. Delaval, A. Girault, M. Pouzet

An effect type system for modular distribution of dataflow programs
Typing rules (declarations)

\[
H \vdash e : s/S, T \\
H \vdash x = e : [s/x]/S, T
\]

\[
H, H_2 \vdash D_1 : H_1/S_1, T_1 \quad H, H_1 \vdash D_2 : H_2/S_2, T_2 \\
H \vdash D_1; D_2 : H_1 \oplus H_2/S_1 \cup S_2, T_1 \cdot T_2
\]

\[
H, x : s \vdash e : s'/S, T \\
H \vdash \text{let node } f(x) = e : [\text{gen}_H(s \leftarrow (S, T) \rightarrow s')/f]/\emptyset, \emptyset
\]
Projection of expressions

\[
H \vdash e : s/S, T \quad T' \vdash e : s < s' \xrightarrow{L} e_2, D_2
\]

\[
H \vdash e : s'/S, T.T' \xrightarrow{L} e_2, D_1; D_2
\]
Projection of expressions

\[
\begin{align*}
H \vdash e : s/S, T & \quad T' \vdash e : s < s' \xrightarrow{L} e_2, D_2 \\
H \vdash e : s'/S, T.T' & \xrightarrow{L} e_2, D_1; D_2
\end{align*}
\]

\[
\begin{align*}
L \in \text{sites}(s) & \quad H \vdash i : s/\text{sites}(S), \emptyset \xrightarrow{L} i, \emptyset \\
L \not\in \text{sites}(s) & \quad H \vdash i : s/\text{sites}(S), \emptyset \xrightarrow{L} d, \emptyset
\end{align*}
\]
Projection of expressions

\[ H \vdash e : s/S, T \quad T' \vdash e : s < s' \quad L \Rightarrow e_2, D_2 \]

\[ H \vdash e : s'/S, T.T' \quad L \Rightarrow e_2, D_1 ; D_2 \]

\[ L \in \text{sites}(s) \]

\[ H \vdash i : s/\text{sites}(S), \emptyset \quad L \Rightarrow i, \emptyset \]

\[ L \notin \text{sites}(s) \]

\[ H \vdash i : s/\text{sites}(S), \emptyset \quad L \Rightarrow d, \emptyset \]

\[ L \in S \]

\[ H \vdash x : s/S, \emptyset \quad L \Rightarrow x, \emptyset \]

\[ L \notin S \]

\[ H \vdash x : s/S, \emptyset \quad L \Rightarrow d, \emptyset \]
Projection of expressions

\[ H \vdash e : s / S, T \quad T' \vdash e : s < s' \xrightarrow{L} e_2, D_2 \]

\[ H \vdash e : s' / S, T.T' \xrightarrow{L} e_2, D_1 ; D_2 \]

\[ L \in \text{sites}(s) \]

\[ H \vdash i : s / \text{sites}(S), \emptyset \xrightarrow{L} i, \emptyset \]

\[ H \vdash x : s / S, \emptyset \xrightarrow{L} x, \emptyset \]

\[ L \notin \text{sites}(s) \]

\[ H \vdash i : s / \text{sites}(S), \emptyset \xrightarrow{L} d, \emptyset \]

\[ H \vdash x : s / S, \emptyset \xrightarrow{L} d, \emptyset \]

\[ H \vdash e_1 : s_1 / S_1, T_1 \xrightarrow{L} e'_1, D_1 \quad H \vdash e_2 : s_2 / S_2, T_2 \xrightarrow{L} e'_2, D_2 \]

\[ H \vdash e_1, e_2 : s_1 \times s_2 / S_1 \cup S_2, T_1.T_2 \xrightarrow{L} (e'_1, e'_2), D_1 ; D_2 \]
Projection of declarations

\[
\begin{array}{c}
H \vdash e : s/S, T \xrightarrow{L} e', D \quad L \in S \\
\hline
H \vdash x = e : [s/x]/S, T \xrightarrow{L} D; x = e'
\end{array}
\]

\[
\begin{array}{c}
L \not\in S \\
H \vdash x = e : [s/x]/S, T \xrightarrow{L} \emptyset
\end{array}
\]

\[
H \vdash D : H'/S, T \xrightarrow{L} D' \quad L \prec_{\mathcal{R}} L'
\]

\[
H \vdash \text{do } D \text{ at } L'/H'/S, T \xrightarrow{L} D'
\]

\[
H, H_2 \vdash D_1 : H_1/S_1, T_1 \xrightarrow{L} D'_1 \quad H, H_1 \vdash D_2 : H_2/S_2, T_2 \xrightarrow{L} D'_2
\]

\[
H \vdash D_1; D_2 : H_1 \oplus H_2/S_1 \cup S_2, T_1 \oplus T_2 \xrightarrow{L} D'_1; D'_2
\]