CSCI.6500/4500 Distributed Computing over the Internet—Programming Distributed Computing Systems (Varela)—Sections 8.1.2, 8.4.4

Instructor: Carlos Varela
Rensselaer Polytechnic Institute
Spring 2021
Distributed and Mobile Systems with Nomadic Pict

We use the Nomadic Pict programming language (Pawel Wojciechowski, Peter Sewell and Benjamin Pierce, 1998), an extension of Pict, to illustrate distributed and mobile systems programming with the $\pi$ calculus.

Nomadic Pict incorporates two main abstractions: *sites* and *agents*.

- Sites denote locations where agents execute.
- Agents can migrate between sites and an agent’s execution is modelled as a Pict process. Channels belong to specific agents.
Nomadic Pict is designed with two levels of communication: location-dependent and location-independent.

- Higher-level location-independent communication can be translated into:
  - lower-level location-dependent communication in many ways
    - obtaining different properties in terms of performance and fault tolerance.
Nomadic Pict Syntax Extension

Proc ::= ... Processes

| Ag | Agent

Ag ::= 

agent \( Id_1 = Proc_1 \) and ... and \( Id_n = Proc_n \) in Proc Create

| migrate to Val Proc Migrate

| iflocal < Val > Val ! Val then Proc else Proc Send

| wait Val ? Abs timeout Val -> Proc Receive

| terminate Kill
Agents

\[ Ag ::= \]

\[ \text{agent } Id_1 = Proc_1 \text{ and } \ldots \text{ and } Id_n = Proc_n \text{ in } Proc \]

\[ | \quad \text{migrate to } Val \ Proc \]

Processes are extended to include *agents* which can be

- created locally and referred to by a name, as well as
- migrated to sites.
Agent Communication

\[
Ag ::= \text{Agents} \\
    | \text{iflocal } < Val > Val ! Val \text{ then } Proc \text{ else } Proc \text{ Send} \\
    | \text{wait } Val ? Abs \text{ timeout Val } \rightarrow \text{Proc} \text{ Receive}
\]

Communication with a local agent is modelled by sending a value over a channel of the agent.

- iflocal \(<a>c!v \text{ then } P \text{ else } Q\)
  
  - Local communication can be tried and alternative processes may execute if it is successful or if it fails.

- wait \(<c?p=P \text{ timeout } t \rightarrow Q)\)
  
  - Waiting for an input on a channel may time out (in seconds) after which a separate process may execute.
Agent Communication

Syntactic sugar may be used for local and remote communication as follows:

\[
\begin{align*}
\langle a \rangle \mathbf{c}!v & \triangleq \text{iflocal } \langle a \rangle \mathbf{c}!v \text{ then } () \text{ else } () \\
\langle a@s \rangle \mathbf{c}!v & \triangleq \text{agent } _= \text{ migrate to s } \langle a \rangle \mathbf{c}!v \text{ in } ()
\end{align*}
\]

Notice that these two forms fail silently if the agent \( a \) is not located in the expected local site or site \( s \) respectively.

A higher-level location-independent communication is used to encode communication irrespective of the current location or migration patterns of an agent:

\[
\mathbf{c}@a!v \triangleq \text{location-independent output to channel } c \text{ at agent } a
\]
Example: An Applet Server

An applet server can be encoded in Nomadic Pict as follows:

```pict
getApplet ?* [a s] =
  agent b =
  migrate to s
  ( ack@a!b | ... )
  in ()
```

The applet server receives a requesting agent `a` and its site `s`, representing a web client and its location respectively. For each request,

- it creates the applet (a new agent named `b`), whose behavior is
  - to migrate to the site `s`,
  - acknowledge its presence by sending its name to the requesting agent `a`, and
  - in parallel execute its code (denoted as `...`)
Mobile Reference Cell Type

A mobile reference cell can be written in Nomadic Pict as an agent that exports three channels:

- a get channel to receive query requests,
- a set channel to receive update requests, and
- a migrate channel to receive migration requests.

```plaintext
type MobRefInt =
[
  get=/[Agent /Int]
  set=/[Agent Int Sig]
  mig=/[Agent Site Sig]
]
```
def mobRefInt (s:Site) : MobRefInt =
   (       
   new get:^[Agent /Int]
   new set:^[Agent Int Sig]
   new mig:^[Agent Site Sig]

   agent refIntAg =
      (       
      new contents:^Int
      run contents!0
      migrate to s
      ( get?^[a:Agent res:/Int] =
         contents?v = (contents!v | res@a!v)
      | set?^[a:Agent v:Int c:Sig] =
         contents?_ = (contents!v | c@a[])
      | mig?^[a:Agent s:Site c:Sig] =
         migrate to s c@a[])
      )
   [       
   get = \^[a:Agent res:/Int] = get@refIntAg![a res]
   set = \^[a:Agent v:Int c:Sig] = set@refIntAg![a v c]
   mig = \^[a:Agent s:Site c:Sig] = mig@refIntAg![a s c]
   ]
)
Mobile Reference Cell in Nomadic Pict

A mobile reference cell \( c \) can be created at site \( s_1 \), and an agent \( a \) can use it and migrate it to another site \( s_2 \) as follows:

```java
val c = (mobRefInt s1)

agent a =

(  
  (c.set a 5);
  (prNL (int.toString (c.get a)));
  (c.mig s2);
  (c.set a 3);
  (prNL (int.toString (c.get a)));

  ()
)
in ()
```