Pi-Calculus Quick Intro

Some History

- In the late 70ties with the grow of distributed systems, some people start to realize that they need a realistic ways of describing communicating systems.

- They create something called process calculi/process algebra.

- Some of these process calculi are:
    - CSP (Tony Hoare): Communicating Sequential Processes
    - CCS (Robin Milner): Calculus of Communicating Systems
    - Pi-calculus (Robin Milner): Extension of CCS, more flexible and support the ability to pass channels as data along other channels. This feature allows you to express process mobility, which in turn allows you to express changes in process structure.

What is Pi-Calculus?

- π -calculus is a model of computation for concurrent systems. The syntax of π-calculus lets you represent processes, parallel composition of processes, synchronous communication between processes through channels, creation of fresh channels and replication of processes.

Process? Channel????

- A process is an abstraction of an independent thread of control.
- A channel is an abstraction of the communication link between two processes. Processes interact with each other by sending and receiving messages over channels.

The Rules

Let P and Q denote processes. Then

- P | Q denotes a process composed of P and Q running in parallel.
- a(x).P denotes a process that waits to read a value x from the channel a and then, having received it, behaves like P.
- $\bar{a}\langle x \rangle.P$ denotes a process that first waits to send the value x along the channel a and then, after x has been accepted by some input process, behaves like P.

The Rules (cont.)

- (νa)P ensures that a is a fresh channel in P. (Read the Greek letter "nu" as "new.")
- !P denotes an infinite number of copies of P, all running in parallel.
- P + Q denotes a process that behaves like either P or Q.
- 0 denotes the inert process that does nothing.

All concurrent behavior that you can imagine would have to be written in terms of just the above constructs.

Simple Example

Suppose you want to model a remote procedure call between a client and a server.

```
int incr(int x) { return x+1; }
```

The Server: $!incr(a, x).\bar{a}\langle x+1\rangle$

The Client: $(\nu a)(\ incr\langle a,\ 17\rangle \mid a(y)\ )$

All Together:

$!incr(a, x).\bar{a}\langle x+1\rangle \mid (\nu\ a)(\ incr\langle a,\ 17\rangle \mid a(y)\ )$

Can you program in π-calculus?

- Yes, but you wouldn’t want to.
- Pi Calculus can be viewed as an assembly language.
- It's good as modeling language. A lot of communication protocols has been expressed using Pi Calculus.

Just for you to know ...

π-calculus is a natural choice for describing concurrent processes that communicate through message passing.

It is not a natural choice for describing abstract data types. It is not a natural choice for describing states with rich or complex data structures.

Pi-Calculus Vs Petri Nets

- Compared in the Process Awareness Information Systems (PAIs) context
- Both Petri Nets and Pi-calc provides solid semantics and strong analysis methods
- In reality only a few cases requires such strong/mathematical foundations.

Pi-Calculus Vs Petri Nets

- Petri Nets are from the early 70ties + ton of extensions
- Pi-Calc comes from the CSS but it was concieved during the early 80ties
- Petri Nets are based on directed graph
- Pi-Calc processes are based on textual descriptions

Pi-Calculus Vs Petri Nets

- Example Petri Net

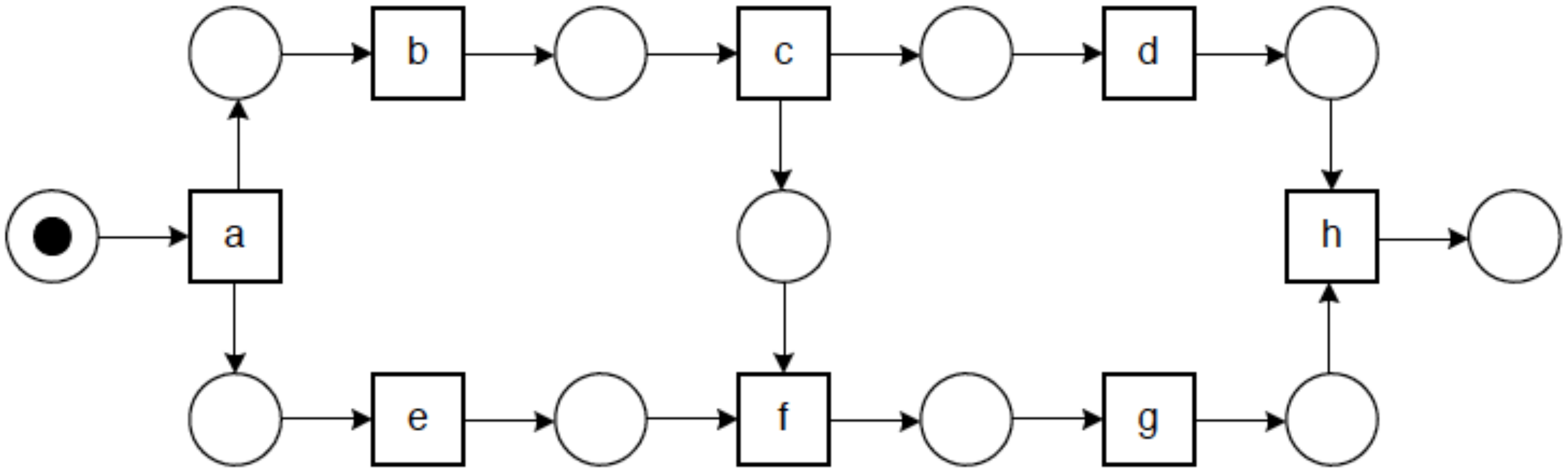

Fig. 1. How to model this in terms of Pi calculus?

- Pi-Calc a.(b.c.d | e.f.g).h WRONG!!

Pi-Calculus Vs Petri Nets

- Petri Nets doesn't have the notion of mobility as foundation
- Petri Nets advantages:
  - Graph representation
  - Expressiveness
  - Improved modeling experience
  - Better readability

Conclusions

- Too formal (mathematical) despite the graphical nature
- State Based instead of Event Based
  - BPMN -> Event Based
  - Pi-Calc -> Event Based
  - Petri Nets -> State Based
- Abundance of Analysis Techniques
  - Both have tons of theory and methods for analysis

My Conclusions

- Formal method of analysis and strong semantics are good, but less useful high level / abstract representations.
- Right now BPMN2 defines a strong syntax and strong behavioral semantics. But it's formality it's not pure mathematical.
- Some sacrifices are allowed for more user friendly/business friendly/realistic and high level representations.