Jobshop Example

- Models assembly line
- Two types of agents (entities) in system:
  - Workers (called “jobbers”)
  - Tools: hammer and mallet
- Workers receive jobs from assembly line (not represented) and use tools to do jobs
- Limited tools; workers compete for access

Jobshop Architecture

Things to Remember

- There could be more jobbers or more tools in the system. Numbers aren’t important; it’s the relationships that are
- A tool may be used by only one jobber at a time
- If two jobbers access a tool at the same time, one is chosen nondeterministically

Components: the Hammer

\[ \text{Hammer} = \text{geth} \cdot \text{Busyhammer} \]
\[ \text{Busyhammer} = \text{puth} \cdot \text{Hammer} \]

\[ \text{equivalent: } \text{Hammer} = \text{geth} \cdot \text{puth} \cdot \text{Hammer} \]

Components: the Mallet

\[ \text{Mallet} = \text{getm} \cdot \text{Busymallet} \]
\[ \text{Busymallet} = \text{putm} \cdot \text{Mallet} \]

\[ \text{equivalent: } \text{Mallet} = \text{getm} \cdot \text{putm} \cdot \text{Mallet} \]

A “Sort”

- … is a set of labels such that all of the operations that an agent can perform in the future are in that set
- \( P : L \) means that all of the actions that agent \( P \) can perform in the future are in \( L \)
- \( L \) may contain other actions, but we usually find the smallest \( L \) that’s useful
**Sorts**

- The smallest sorts for some of the agents in the Jobshop:

  - Hammer: \{geth, putm\}
  - Mallet: \{getm, putm\}
  - Jobshop: \{in, out\}

**The Jobber**

- The jobber has the following sort:

  \( \text{Jobber} : \{\text{in, out, geth, putm, getm, putm}\} \)

**The Jobber’s Description**

\[
\text{Jobber} = \text{in}(\text{job}), \text{Start}(\text{job}) \\
\text{Start}(\text{job}) = \text{if } \text{easy}(\text{job}) \text{ then } \text{Finish}(\text{job}) \\
\text{else if } \text{hard}(\text{job}) \text{ then } \text{Usehammer}(\text{job}) \\
\text{else } \text{Usetool}(\text{job}) \\
\text{Usetool}(\text{job}) = \text{Usehammer}(\text{job}) + \text{Usemallet}(\text{job}) \\
\text{Usehammer}(\text{job}) = \text{geth}.\text{putm}.\text{Finish}(\text{job}) \\
\text{Usemallet}(\text{job}) = \text{getm}.\text{putm}.\text{Finish}(\text{job}) \\
\text{Finish}(\text{job}) = \text{out}(\text{done}(\text{job})).\text{Jobber}
\]

**Linking Components Together**

- Linking a Jobber and a Hammer:

  \( \text{Jobber} \mid \text{Hammer} \)

**Encapsulating Components**

- Encapsulating interactions between a Jobber and a Hammer:

  \( (\text{Jobber} \mid \text{Hammer})\{\text{geth, putm}\} \)

**Jobshop Declaration**

\[
\text{Jobshop} = (\text{Jobber} \mid \text{Jobber} \mid \text{Hammer} \mid \text{Mallet})L \\
\text{where } L = \{\text{geth, putm, getm, putm}\}
\]

- which is equivalent to:

  \( ((\text{Jobber} \mid \text{Jobber} \mid \text{Hammer})\{\text{geth, putm}\} \mid \text{Mallet})\{\text{getm, putm}\} \)

- and potentially many other forms...
Additional Notes on Sorts

Jobshop, sort of...

Jobshop : \{in, out\}

- If \( C = (A | B) \) and \( A : X \) and \( B : Y \), then
  \( C : X \cup Y \)

- If \( C = (A | B) \setminus Z \) and \( A : X \) and \( B : Y \), then
  \( C : X \cup Y - (\{Z\} \cup (\bigcup_{z \in Z}) \) \)

A generic “reusable” tool...

Define a semaphore agent

“Sem” with put and get operations (P and V)

Sem = get put Sem

We can “reuse” the Sem definition to define a Hammer and a Mallet:

Hammer = Sem[geth/get, puth/put]
Mallet = Sem[getm/get, putm/put]

Equivalence of systems

Define a “Strongjobber” that can do the same job as a normal Jobber but with his hands, without tools:

Strongjobber = in(job).out(done(job)).Strongjobber

One can then claim that

Jobshop = Strongjobber | Strongjobber

because both can complete two jobs in parallel. The use of tools in Jobshop is encapsulated, so we can’t see it; therefore it’s not significant when determining system equivalence.