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

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

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

\[ \text{Mallet} = \text{getm.Busymallet} \]

\[ \text{Busymallet} = \text{putm.Mallet} \]

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

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:

  \( \text{Hammer} : \{ \text{geth, puth} \} \)

  \( \text{Mallet} : \{ \text{getm, putm} \} \)

  \( \text{Jobshop} : \{ \text{in, out} \} \)

The Jobber

The jobber has the following sort:

\( \text{Jobber} : \{ \text{in, out, geth, puth, getm, putm} \} \)
The Jobber’s Description

\[ \text{Jobber} = \text{in}(job) \cdot \text{Start}(job) \]

\[ \text{Start}(job) = \begin{cases} \text{Finish}(job) & \text{if easy}(job) \\ \text{Usehammer}(job) & \text{if hard}(job) \\ \text{Usetool}(job) & \text{else} \end{cases} \]

\[ \text{Usetool}(job) = \text{Usehammer}(job) + \text{Usemallet}(job) \]

\[ \text{Usehammer}(job) = \text{geth} \cdot \text{puth} \cdot \text{Finish}(job) \]

\[ \text{Usemallet}(job) = \text{getm} \cdot \text{putm} \cdot \text{Finish}(job) \]

\[ \text{Finish}(job) = \text{out}(\text{done}(job)) \cdot \text{Jobber} \]

Linking Components Together

Linking a Jobber and a Hammer:

\[ \text{Jobber} \mid \text{Hammer} \]

\[ \begin{array}{c}
\text{Jobber} \\
\hline
\text{Hammer} \\
\end{array} \]

\[ \text{in} \quad \text{out} \]

\[ \text{geth} \quad \text{puth} \]

\[ \text{getm} \quad \text{putm} \]
Encapsulating Components

Encapsulating interactions between a Jobber and a Hammer:

\[(\text{Jobber} \mid \text{Hammer}) \backslash \{\text{geth, puth}\}\]

Jobshop Declaration

\[\text{Jobshop} = (\text{Jobber} \mid \text{Jobber} \mid \text{Hammer} \mid \text{Mallet}) \backslash L\]

where \(L = \{\text{geth, puth, getm, putm}\}\)

which is equivalent to:

\[((\text{Jobber} \mid \text{Jobber} \mid \text{Hammer}) \backslash \{\text{geth, puth}\} \mid \text{Mallet}) \backslash \{\text{getm, putm}\}\]

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

Jobshop, sort of…

\[ \text{Jobshop} : \{\text{in, out}\} \]

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

A generic “reusable” tool…

Define a semaphore agent “Sem” with put and get operations (P and V)

\[ \text{Sem} = \text{get}.\text{put}.\text{Sem} \]

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

\[ \text{Hammer} = \text{Sem}[\text{geth}/\text{get}, \text{puth}/\text{put}] \]
\[ \text{Mallet} = \text{Sem}[\text{getm}/\text{get}, \text{putm}/\text{put}] \]
Equivalence of systems

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

\[ Strongjobber = \text{in}(job).\overline{\text{out}}(\text{done}(job)).\text{Strongjobber} \]

One can then claim that

\[ \text{Jobshop} = Strongjobber \mid Strongjobber \]

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.