# A Data Flow Modeling Language for WS Composition

Annapaola Marconi
marconi@itc.it

February 23, 2006 - ASTRO Seminar

# Outline

1. Introduction
   - Aim
   - State of the art
2. Data Flow Language
   - Syntax
   - Semantics
   - VTA Case Study
3. Future Works

# Language Aim

- Specify how messages sent to component services must be obtained from messages received from component services. e.g. :
  - how several input messages must be combined to obtain an output message,
  - whether an input message can be used several times or just once
  - whether all messages received must be processed and sent or not

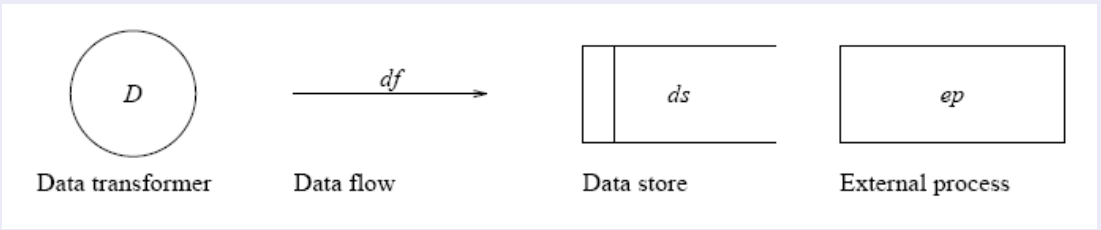
# Language Aim

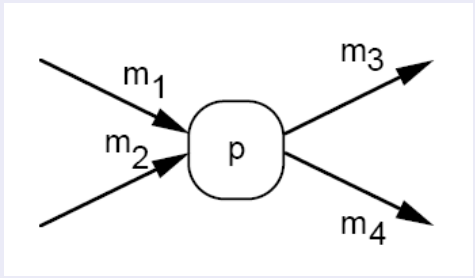
- Specify how messages sent to component services must be obtained from messages received from component services. e.g. :
  - how several input messages must be combined to obtain an output message,
  - whether an input message can be used several times or just once
  - whether all messages received must be processed and sent or not

- Specify message precedences and other requirements related to the order in which messages must be sent is not an issue of this language.

# Existing Modeling Languages

## Data Flow Diagrams



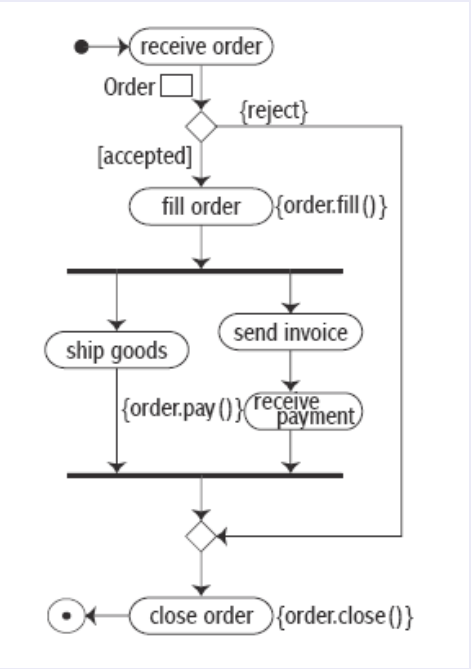




- ambiguous semantics
- several formalization attempts, e.g.
  - *Bruza et al. , 1993*: DFD + activation possibilities $\longrightarrow$ Petri Nets
  - *Larsen et al. , 1993*: DFD + firing rules $\longrightarrow$ VDM (Vienna Development Method)

# Existing Modeling Languages

## UML 2.0 Activity Diagrams

- control flow + data flow
- not clear and precise semantics
  - *Storrle et al. , 2005*:
    AD $\longrightarrow$ colored Petri-nets



# Outline

1 Introduction
- Aim
- State of the art

2 Data Flow Language
- Syntax
- Semantics
- VTA Case Study

3 Future Works

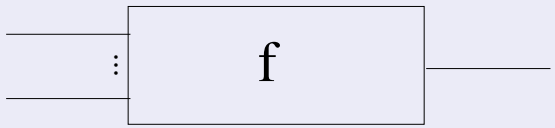
# Language Components

## Identity


Syntax: Has one input and one output channel of the same type.
Intuitive Semantics: It forwards data received from the input channel to the output channel.

# Language Components

## Operation

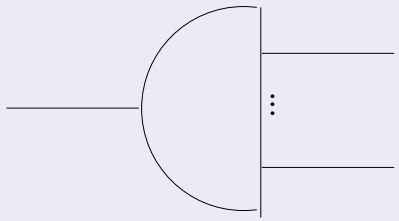



Syntax: It is related to a function definition, has one input channel for each function parameter (the type of each channel is the same of the parameter) and only one output channel corresponding to the function result.

Intuitive Semantics: When it receives data on all the input channels, it computes the result and forwards it to the output channel.

# Language Components

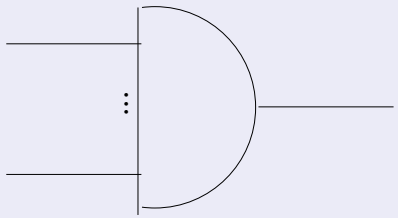
## Fork



Syntax: Has one input channel and as many output channels as necessary. All channels are of the same type.
Intuitive Semantics: It forwards data received on the input channel to all the output channels.

# Language Components

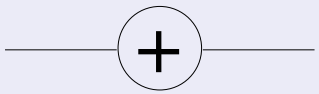
## Merge



Syntax: Has only one output channel and as many input channels as necessary. All channels are of the same type.
Intuitive Semantics: It forwards data received on some input channel to the output channel. It preserves the temporal order of data arriving on input channels (if it receives data on two or more input channels at the same time, the order is nondeterministic).

# Language Components

## Cloner



Syntax: Has one input channel and one output channel and they are of the same type.
Intuitive Semantics: It forwards, once or more times, data received from the input channel to the output channel.

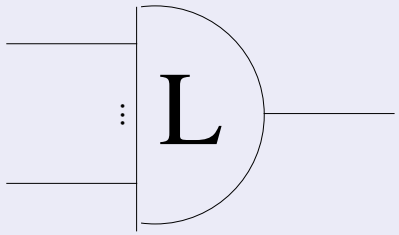
# Language Components

## Filter


Syntax: Has one input channel and one output channel and they are of the same type.
Intuitive Semantics: When it receives data on the input channel it either forwards it to the output channel or discards it.

# Language Components

## Last




Syntax: Has only one output channel and as many input channels as necessary. All channels are of the same type.
Intuitive Semantics: It forwards to the output channel at most one data: the last data received on the input channels. It discards all other data received.

# Language Components

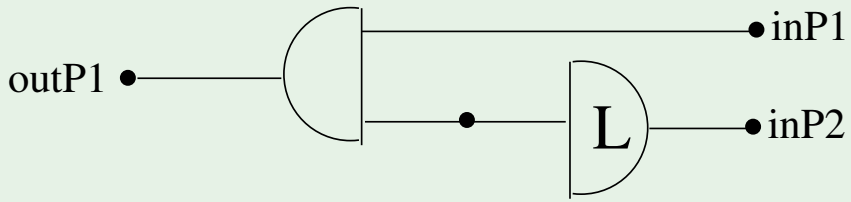
## Connection Node

Is used to compose different language components. Can be external or internal:

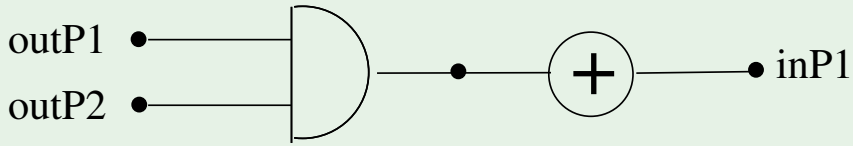
- an external connection node is associated to an output (or an input) external port and can be connected to an input (or an output) channel of the same type;
- an internal connection node connects an output channel to an input channel of the same type.

# Examples

## example1




## example2



# Outline

1 Introduction
- Aim
- State of the art

2 Data Flow Language
- Syntax
- Semantics
- VTA Case Study

3 Future Works

# Semantic Domain

## Data Net

The diagram obtained by modeling the data flow requirements with the specified language is called data net.

# Semantic Domain

## Data Net

The diagram obtained by modeling the data flow requirements with the specified language is called data net.

## Event

An event $e$ is a couple $(n, d)$, where $n$ is a connection node and $d$ is a data item, and models the fact that the data item $d$ passes through the connection node $n$;

- $node(e)$ denotes the node of event $e$
- $data(e)$ denotes the data of event $e$

# Semantic Domain

## Execution

An execution $\rho$ of a data net is a finite sequence of events $e_0, ..e_n$

# Semantic Domain

## Execution

An execution $\rho$ of a data net is a finite sequence of events $e_0, ..e_n$

- Given an execution $\rho$ we define it's projection on a set of connection nodes $N$, and denote it with $\Pi_N(\rho)$, the ordered sequence $x_0, ..x_m$ representing the indexes in $\rho$ of all events whose node is in $N$.

# Semantic Domain

## Execution

An execution $\rho$ of a data net is a finite sequence of events $e_0, ..e_n$

- Given an execution $\rho$ we define it's projection on a set of connection nodes $N$, and denote it with $\Pi_N(\rho)$, the ordered sequence $x_0, ..x_m$ representing the indexes in $\rho$ of all events whose node is in $N$.
- Given an execution $\rho$, we denote with $event(\rho)[i]$ the function that returns the $i$-th event in execution $\rho$.

# Semantic Mapping

**Accepting execution**

An execution $\rho = e_0, .., e_n$ is accepted by a data net if it satisfies all the following properties:

# Semantic Mapping

## Accepting execution

An execution $\rho = e_0, .., e_n$ is accepted by a data net if it satisfies all the following properties:

## Property 1

for each Identity element in the data net


where $\Pi_{\{a\}}(\rho) = x_0, ..x_k$ and $\Pi_{\{b\}}(\rho) = y_0, ..y_m$,

- $k = m$
- $data(e_{x_i}) = data(e_{y_i})$, $0 \leq i \leq k$
- $y_i > x_i$, $0 \leq i \leq k$

# Semantic Mapping

## Accepting execution

An execution $\rho = e_0, .., e_n$ is accepted by a data net if it satisfies all the following properties:

## Property 1

for each Identity element in the data net


where $\Pi_{\{a\}}(\rho) = x_0, ..x_k$ and $\Pi_{\{b\}}(\rho) = y_0, ..y_m$,

- $k = m$
- $data(e_{x_i}) = data(e_{y_i})$, $0 \leq i \leq k$
- $y_i > x_i$, $0 \leq i \leq k$

Accepting

- (a,1),(b,1),(a,2),(b,2)
- (a,1),(a,2),(b,1),(b,2)
- (a,1),(a,1),(b,1),(a,2),(b,1),(b,2)

NON-accepting

- (a,1),(a,2),(b,1)
- (a,1),(b,1),(a,2),(b,1)
- (a,1),(a,1),(b,1),(a,2),(b,2),(b,1)

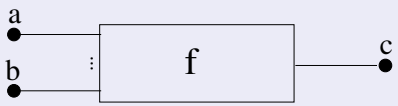
# Semantic Mapping

## Accepting execution

An execution $\rho = e_0, .., e_n$ is accepted by a data net if it satisfies all the following properties:

## Property 2

for each Operation element in the data net




where $\Pi_{\{a\}}(\rho) = x_0, ..x_m$, $\Pi_{\{b\}}(\rho) = y_0, ..y_k$, and $\Pi_{\{c\}}(\rho) = z_0, ..z_p$,

- $p = min(m, k)$
- $data(e_{z_i}) = f(data(e_{x_i}), data(e_{y_i}))$, $0 \leq i \leq p$
- $z_i > max(x_i, y_i)$, $0 \leq i \leq p$

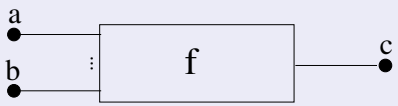
# Semantic Mapping

## Accepting execution

An execution $\rho = e_0, .., e_n$ is accepted by a data net if it satisfies all the following properties:

## Property 2

for each Operation element in the data net




where $\Pi_{\{a\}}(\rho) = x_0, ..x_m$, $\Pi_{\{b\}}(\rho) = y_0, ..y_k$, and $\Pi_{\{c\}}(\rho) = z_0, ..z_p$,

- $p = min(m, k)$
- $data(e_{z_i}) = f(data(e_{x_i}), data(e_{y_i}))$, $0 \leq i \leq p$
- $z_i > max(x_i, y_i)$, $0 \leq i \leq p$

Accepting (f=sum(x,y))

- (a,1),(a,2),(b,1),(b,2),(c,2),(c,4)
- (a,1),(a,2),(b,1),(c,2),(b,0),(c,2)
- (a,1),(a,2),(b,2),(c,3)

NON-accepting

- (a,1),(c,2),(b,1)
- (a,1),(a,0),(b,1),(b,2),(c,1),(c,3)
- (a,1),(b,1),(c,2),(a,2),(b,0)

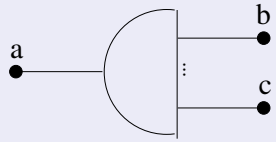
# Semantic Mapping

## Accepting execution

An execution $\rho = e_0, .., e_n$ is accepted by a data net if it satisfies all the following properties:

## Property 3

for each Fork element in the data net




where $\Pi_{\{a\}}(\rho) = x_0, ..x_m$, $\Pi_{\{b\}}(\rho) = y_0, ..y_k$, and $\Pi_{\{c\}}(\rho) = z_0, ..z_p$,

- $p = m = k$
- $data(e_{z_i}) = data(e_{x_i}) = data(e_{y_i})$, $0 \leq i \leq p$
- $y_i, z_i > x_i$, $0 \leq i \leq p$

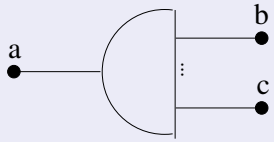
# Semantic Mapping

## Accepting execution

An execution $\rho = e_0, .., e_n$ is accepted by a data net if it satisfies all the following properties:

## Property 3

for each Fork element in the data net




where $\Pi_{\{a\}}(\rho) = x_0, ..x_m$, $\Pi_{\{b\}}(\rho) = y_0, ..y_k$, and $\Pi_{\{c\}}(\rho) = z_0, ..z_p$,

- $p = m = k$
- $data(e_{z_i}) = data(e_{x_i}) = data(e_{y_i})$, $0 \leq i \leq p$
- $y_i, z_i > x_i$, $0 \leq i \leq p$

Accepting

- (a,1),(b,1),(c,1),(a,0),(b,0),(c,0)
- (a,1),(b,1),(a,0),(c,1),(b,0),(c,0)

NON-accepting

- (a,1),(c,1)
- (a,1),(a,0),(b,0),(b,1),(c,1),(c,0)
- (a,1),(b,1),(c,1),(c,2),(a,2),(b,2)

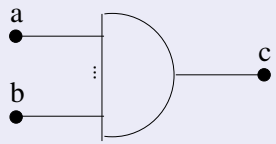
# Semantic Mapping

## Accepting execution

An execution $\rho = e_0, .., e_n$ is accepted by a data net if it satisfies all the following properties:

## Property 4

for each Merge element in the data net




where $\Pi_{\{a,b\}}(\rho) = x_0, ..x_m$ and $\Pi_{\{c\}}(\rho) = y_0, ..y_k$,

- $m = k$
- $data(e_{x_i}) = data(e_{y_i})$, $0 \leq i \leq p$
- $y_i > x_i$, $0 \leq i \leq p$

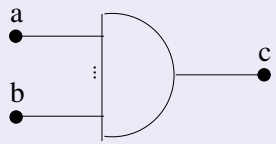
# Semantic Mapping

## Accepting execution

An execution $\rho = e_0, .., e_n$ is accepted by a data net if it satisfies all the following properties:

## Property 4

for each Merge element in the data net




where $\Pi_{\{a,b\}}(\rho) = x_0, ..x_m$ and $\Pi_{\{c\}}(\rho) = y_0, ..y_k$,

- $m = k$
- $data(e_{x_i}) = data(e_{y_i})$, $0 \leq i \leq p$
- $y_i > x_i$, $0 \leq i \leq p$

Accepting

- (a,1),(a,2),(c,1),(b,1),(c,2),(c,1)
- (a,1),(b,0),(c,1),(a,2),(c,0),(c,2)

NON-accepting

- (a,1),(b,0),(c,1)
- (a,1),(b,0),(c,0),(c,1)
- (c,1)

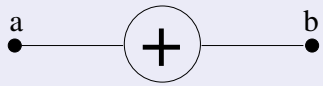
# Semantic Mapping

## Accepting execution

An execution $\rho = e_0, .., e_n$ is accepted by a data net if it satisfies all the following properties:

## Property 5

for each Cloner element in the data net




where $\Pi_{\{a\}}(\rho) = x_0, ..x_m$ and $\Pi_{\{b\}}(\rho) = y_0, ..y_p$,

- $p \geq m$
- there exist $j^0, ..j^{m+1}$ s.t. $j^0 = 0$, $j^{m+1} = p + 1$, $j^i > j^{i-1}$ and $data(e_{x_i}) = data(e_{y_k})$, $j^i \leq k < j^{i+1}$

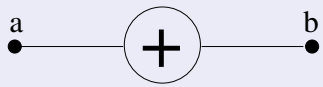
# Semantic Mapping

## Accepting execution

An execution $\rho = e_0, .., e_n$ is accepted by a data net if it satisfies all the following properties:

## Property 5

for each Cloner element in the data net




where $\Pi_{\{a\}}(\rho) = x_0, ..x_m$ and $\Pi_{\{b\}}(\rho) = y_0, ..y_p$,

- $p \geq m$
- there exist $j^0, ..j^{m+1}$ s.t. $j^0 = 0$, $j^{m+1} = p + 1$, $j^i > j^{i-1}$ and $data(e_{x_i}) = data(e_{y_k})$, $j^i \leq k < j^{i+1}$

Accepting

- (a,1),(b,1)
- (a,1),(a,2),(b,1),(b,1),(b,2)
- (a,1),(b,1),(a,2),(b,1),(b,1),(b,2)

NON-accepting

- (a,1),(a,2),(b,1),(b,2),(b,1)
- (a,1),(b,1),(b,2),(a,2)
- (a,1),(b,1),(b,1),(a,2),(b,1)

# Semantic Mapping

## Accepting execution

An execution $\rho = e_0, .., e_n$ is accepted by a data net if it satisfies all the following properties:

## Property 6

for each Filter element in the data net


where $\Pi_{\{a\}}(\rho) = x_0, ..x_m$ and $\Pi_{\{b\}}(\rho) = y_0, ..y_p$,

- $p \leq m$
- there exist $j^0, ..j^p$ s.t. $j^i > j^{i-1}$ and $data(e_{x_{j^i}}) = data(e_{y_i})$, $0 \leq i \leq p$

# Semantic Mapping

## Accepting execution

An execution $\rho = e_0, .., e_n$ is accepted by a data net if it satisfies all the following properties:

## Property 6

for each Filter element in the data net


where $\Pi_{\{a\}}(\rho) = x_0, ..x_m$ and $\Pi_{\{b\}}(\rho) = y_0, ..y_p$,

- $p \leq m$
- there exist $j^0, ..j^p$ s.t. $j^i > j^{i-1}$ and $data(e_{x_{j^i}}) = data(e_{y_i})$, $0 \leq i \leq p$

Accepting

- (a,1),(a,2),(b,1),(b,2)
- (a,1),(a,2),(b,1)
- (a,1),(a,0),(a,2),(b,2)

NON-accepting

- (a,1),(a,2),(b,1),(b,1)
- (a,1),(b,0),(a,0)

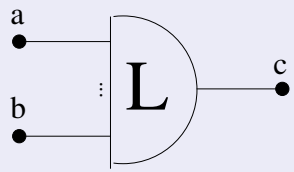
# Semantic Mapping

## Accepting execution

An execution $\rho = e_0, .., e_n$ is accepted by a data net if it satisfies all the following properties:

## Property 7

for each Last element in the data net




where $\Pi_{\{a,b\}}(\rho) = x_0, ..x_m$ and $\Pi_{\{c\}}(\rho) = y_0, ..y_k$,

- $k = 0$
- $y_0 > x_m$
- $data(e_{y_0}) = data(e_{x_m})$

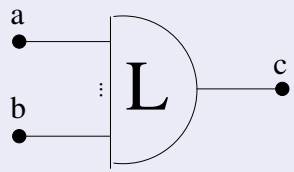
# Semantic Mapping

## Accepting execution

An execution $\rho = e_0, .., e_n$ is accepted by a data net if it satisfies all the following properties:

## Property 7

for each Last element in the data net




where $\Pi_{\{a,b\}}(\rho) = x_0, ..x_m$ and $\Pi_{\{c\}}(\rho) = y_0, ..y_k$,

- $k = 0$
- $y_0 > x_m$
- $data(e_{y_0}) = data(e_{x_m})$

Accepting

- (a,1),(a,0),(b,1),(b,2),(c,2)
- (a,1),(a,2),(b,1),(c,1)

NON-accepting

- (a,1),(c,1),(b,1),(c,1)
- (a,1),(a,0),(b,1)
- (a,1),(b,0),(c,1)

# WS Composition Model

## State Transition System (STS)

A state transition system $\Sigma$ is a tuple $\langle \mathcal{S}, \mathcal{S}^0, \mathcal{I}, \mathcal{O}, \mathcal{R}, \mathcal{L} \rangle$ where:

- $\mathcal{S}$ is the finite set of states;
- $\mathcal{S}^0 \subseteq \mathcal{S}$ is the set of initial states;
- $\mathcal{I}$ is the finite set of input actions;
- $\mathcal{O}$ is the finite set of output actions;
- $\mathcal{R} \subseteq \mathcal{S} \times (\mathcal{I} \cup \mathcal{O} \cup \{\tau\}) \times \mathcal{S}$ is the transition relation;
- $\mathcal{L} : \mathcal{S} \rightarrow 2^{\mathcal{P}rop}$ is the labeling function.

# WS Composition Model

## State Transition System (STS)

A state transition system $\Sigma$ is a tuple $\langle \mathcal{S}, \mathcal{S}^0, \mathcal{I}, \mathcal{O}, \mathcal{R}, \mathcal{L} \rangle$ where:

- $\mathcal{S}$ is the finite set of states;
- $\mathcal{S}^0 \subseteq \mathcal{S}$ is the set of initial states;
- $\mathcal{I}$ is the finite set of input actions;
- $\mathcal{O}$ is the finite set of output actions;
- $\mathcal{R} \subseteq \mathcal{S} \times (\mathcal{I} \cup \mathcal{O} \cup \{\tau\}) \times \mathcal{S}$ is the transition relation;
- $\mathcal{L} : \mathcal{S} \rightarrow 2^{\mathcal{P}rop}$ is the labeling function.

- $\mathcal{I} = P^{in} \times V$ and $\mathcal{O} = P^{out} \times V$;
  - $P^{in} = \bigcup P_i^{in}$ and $P^{out} = \bigcup P_i^{out}$
  - $a = (p, v)$ models the receiving/sending of a value $v$ on a particular service port $p$
  - if $a = (p, v) \in \mathcal{I} \cup \mathcal{O}$, then we write $port(a)$ for the port component $p$, and $value(a)$ for the value component $v$, of $a$

# WS Composition Model

## Recall some def..

- $a \in (\mathcal{I} \cup \mathcal{O} \cup \{\tau\})$ is applicable on a state $s \in \mathcal{S}$ if there exists a state $s' \in \mathcal{S}$ s.t. $(s, a, s') \in \mathcal{R}$;
- $s \in \mathcal{S}$ is final if no action $a \in (\mathcal{I} \cup \mathcal{O} \cup \{\tau\})$ is applicable in $s$;
- a (finite) word $w$ is an element of $\{\mathcal{I} \cup \mathcal{O} \cup \{\tau\}\}^*$, i.e., a finite sequence $a^0, .., a^n$ of actions.
- a run $\sigma$ of $\Sigma$ on a (finite) word $w{=}a^0, .., a^{n-1}$ is a sequence $s^0, .., s^n$ of states in $\mathcal{S}$ such that, $s^0 \in \mathcal{S}^0$, and $\langle s^i, a^i, s^{i+1} \rangle \in \mathcal{R}$, $0 \leq i < n$; the run $\sigma$ is accepting if $s^n$ is a final state.
- $w$ is accepted by $\Sigma$ if $\Sigma$ has an accepting run on $w$.

# WS Composition Model

## Data Net Satisfiability

Let $\Sigma$ be a STS and $\mathcal{D}$ a data net. We say that $\Sigma$ satisfies $\mathcal{D}$, denoted with $\Sigma \models \mathcal{D}$, when for each word $w$ accepted by $\Sigma$, there exists an execution $\rho$ accepted by $\mathcal{D}$ such that:

- $n \leq m$;
- there exist $j^0, ..j^n$, $j^i > j^{i-1}$, s.t. $data(e_{j^i}) = value(a_i)$ and $extport(node(e_i)) = port(a_i)$;
- the number of output actions in $w$ is equal to the number of events in $event(\rho)[\Pi_{N_O}(\rho)]$;

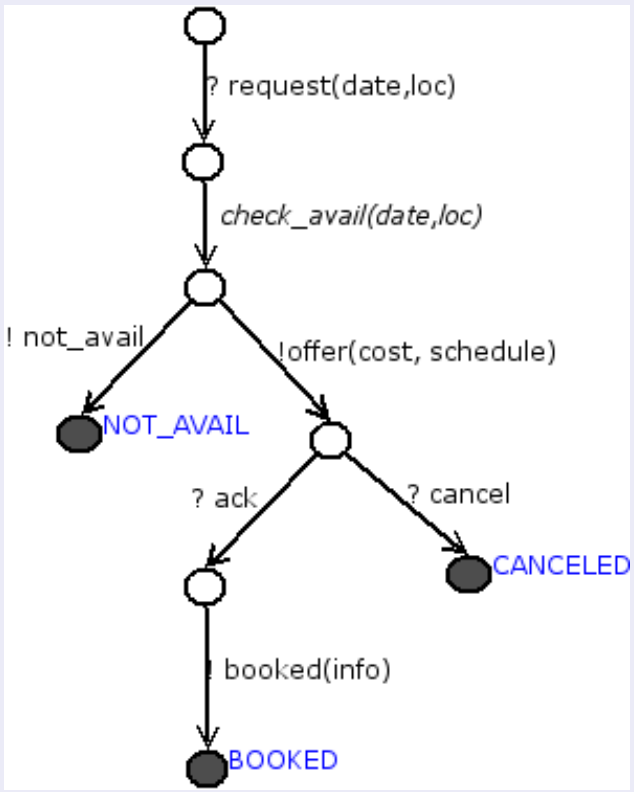
where $a_0, ..a_n = w|_{\mathcal{I} \cup \mathcal{O}}$ and $e_o, ..e_m = event(\rho)[\Pi_{N_{ext}}(\rho)]$.

# Outline

1. Introduction
   - Aim
   - State of the art
2. Data Flow Language
   - Syntax
   - Semantics
   - VTA Case Study
3. Future Works

# The VTA Case Study

## Flight Service Protocol




## Flight Input Ports

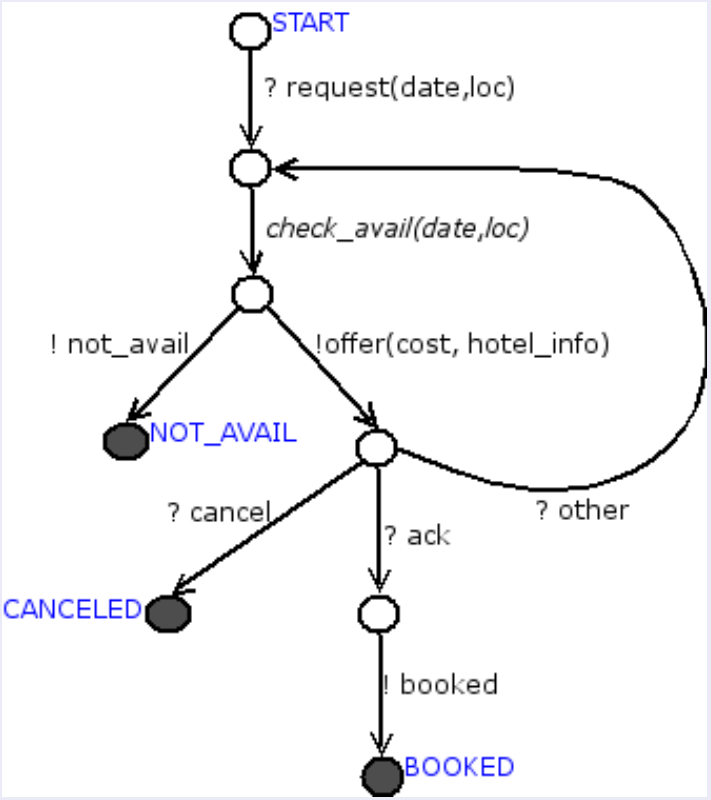
f.request.date
f.request.loc

## Flight Output Ports

f.offer.cost
f.offer.schedule
f.booked.info

# The VTA Case Study

## Hotel Service Protocol




## Hotel Input Ports

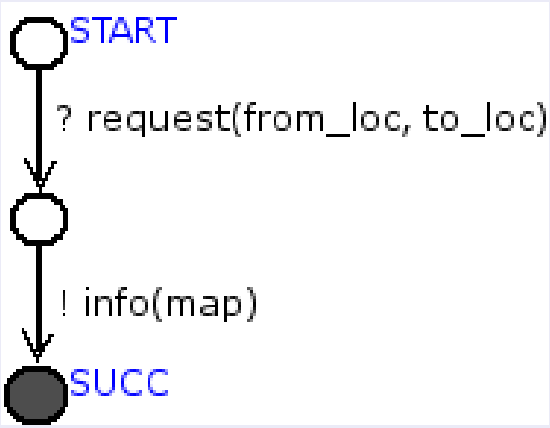
h.request.date
h.request.loc

## Hotel Output Ports

h.offer.cost
h.offer.hotel_info

# The VTA Case Study

## AllMaps Service Protocol




## AllMaps Input Ports

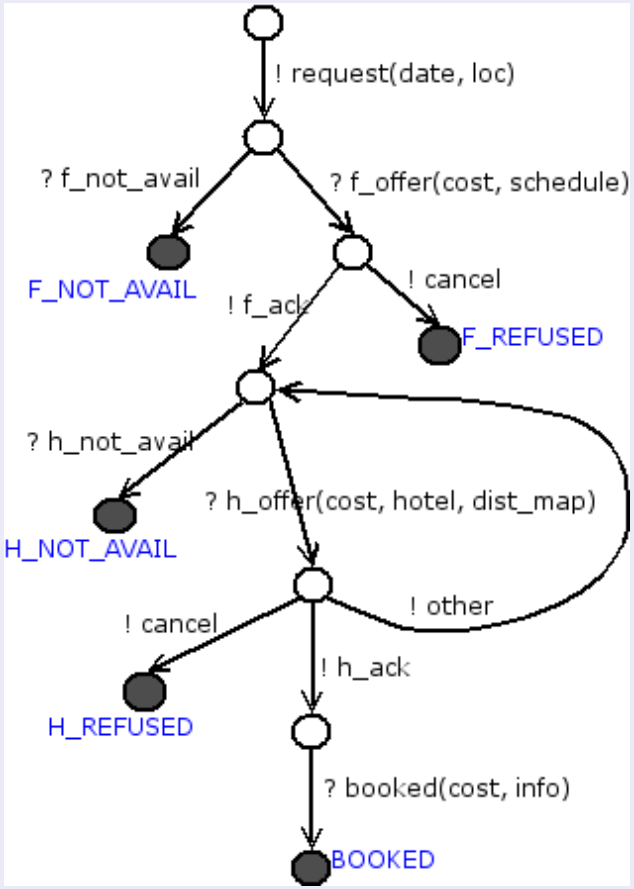
m.request.from_loc
m.request.to_loc

## AllMaps Output Ports

m.info.map

# The VTA Case Study

## Customer Service Protocol




## Customer Input Ports

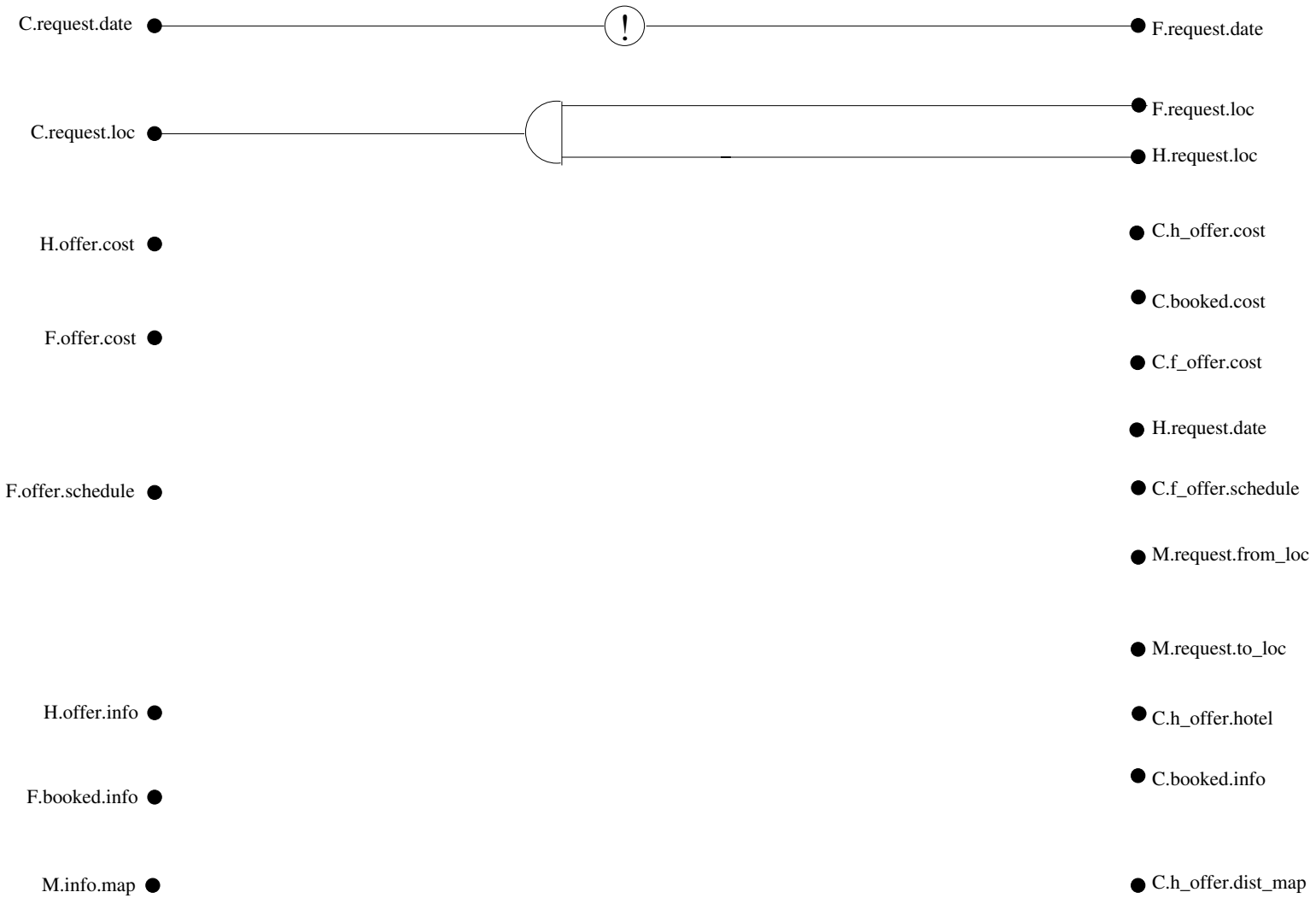
c.f_offer.cost
c.f_offer.schedule
c.h_offer.cost
c.f_offer.hotel
c.f_offer.dist_map
c.booked.cost
c.booked.info

## Customer Output Ports

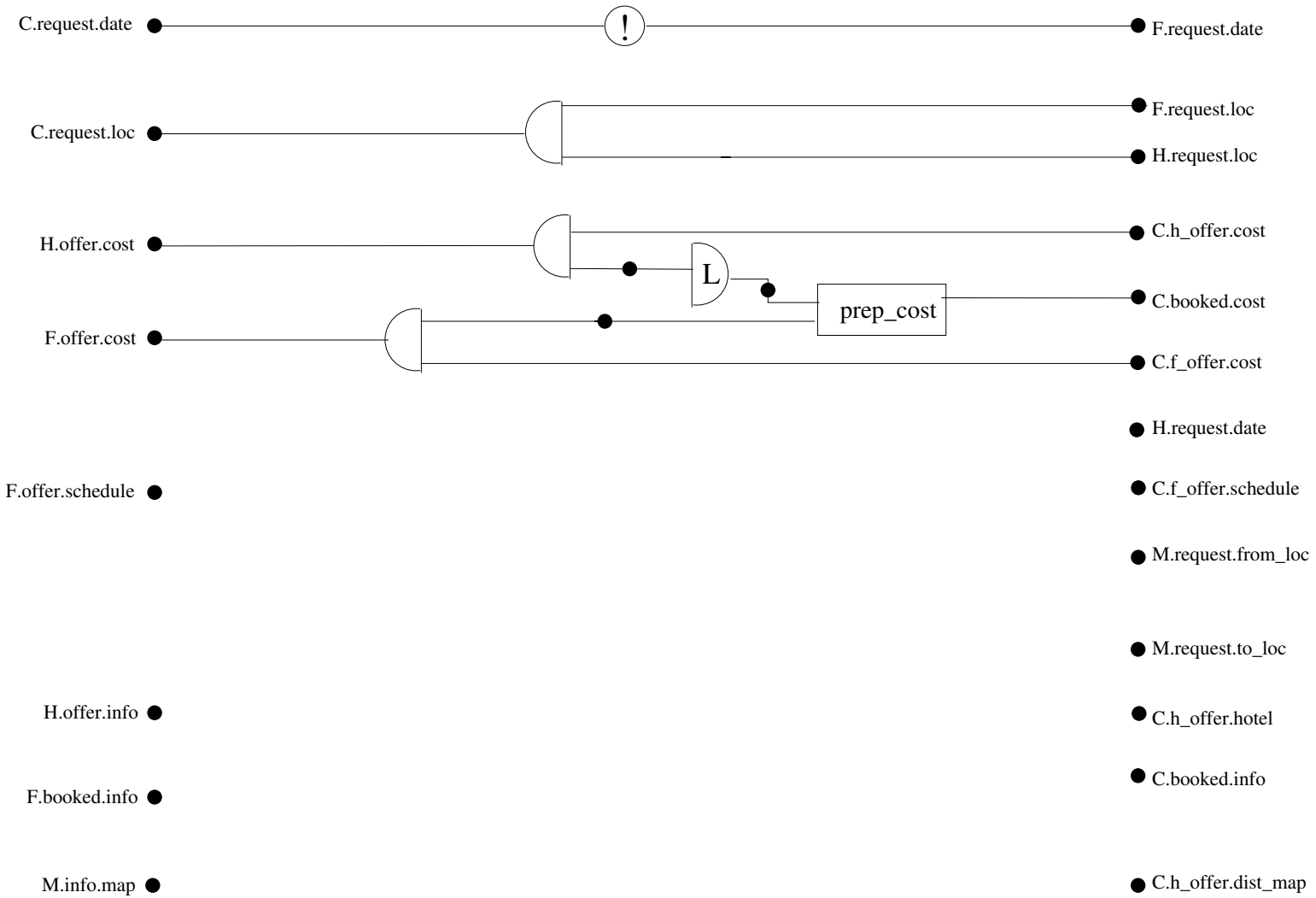
c.request.date
c.request.loc

# The VTA Case Study

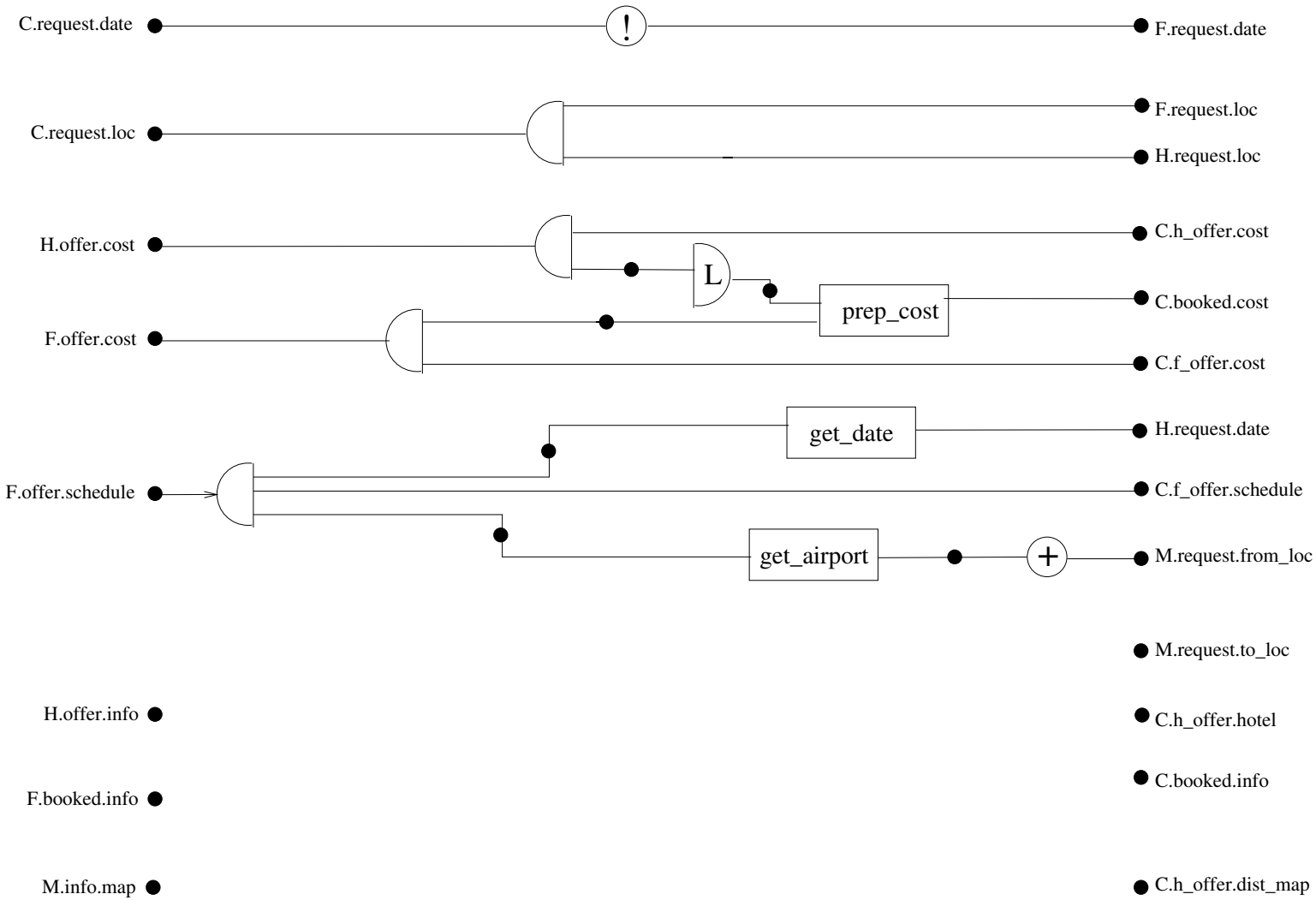
# The VTA Case Study



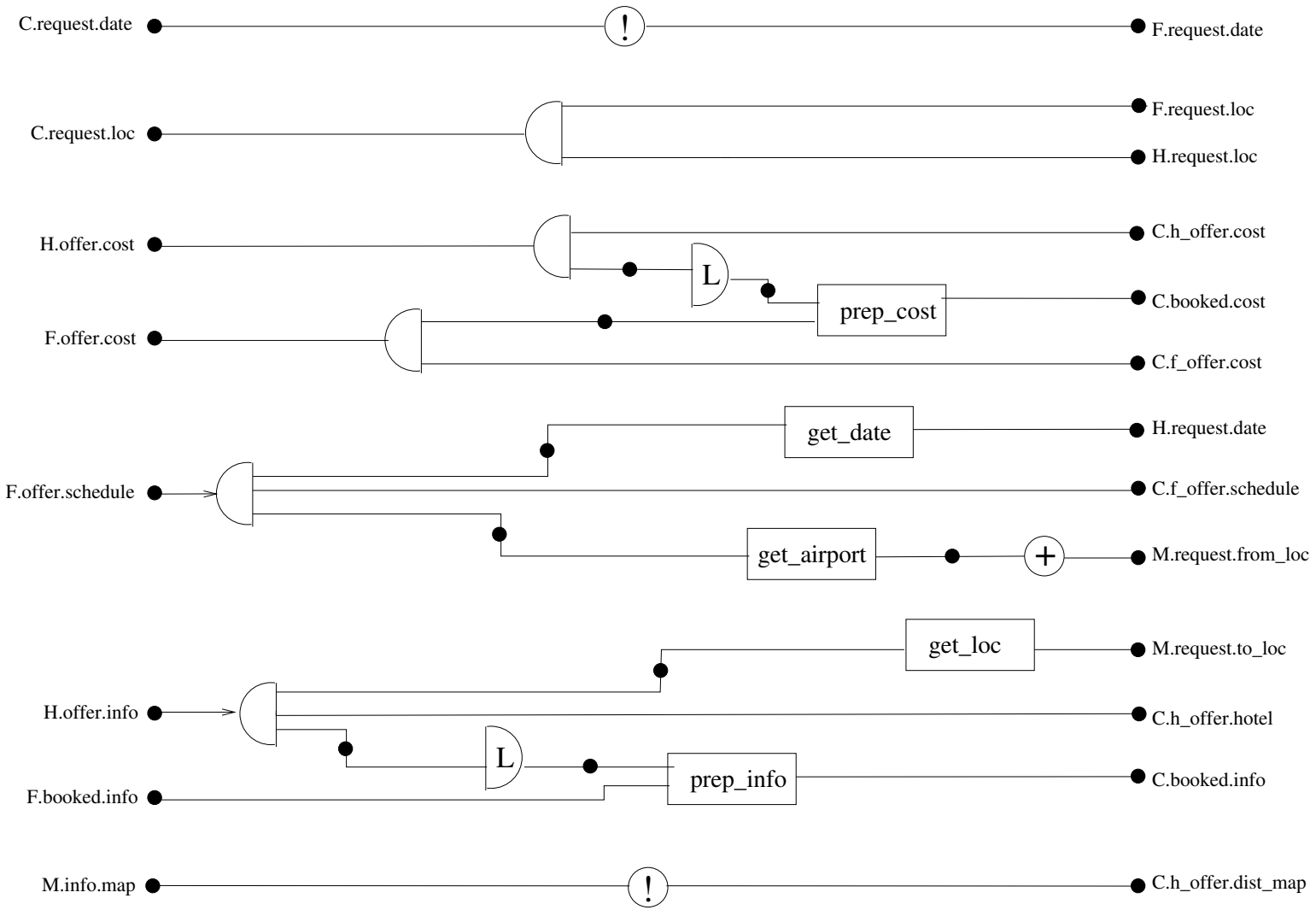
# The VTA Case Study



# The VTA Case Study



# The VTA Case Study



# Future Works

- extend the composition framework to support the specification of data flow requirements in the composition goal
- develop a tool supporting the specification of data flow requirements
- extend the modeling language to support the specification of control flow requirements and complex termination conditions