



2011 ... Paphos, Cyprus

Formal Modeling of RESTful Systems Using Finite-State Machines

Ivan Zuzak, Ivan Budiselic, Goran Delac
School of Electrical Engineering and Computing,
University of Zagreb, Zagreb, Croatia



UNIVERSITY OF ZAGREB
FACULTY OF
ELECTRICAL
ENGINEERING
AND COMPUTING



Ivan Zuzak

- Ph.D student, University of Zagreb
 - Consumer Computing Laboratory



Consumer Computing
Laboratory

- **Geppeto** <http://www.geppeto.fer.hr>
 - Consumer programming methodology
 - Architectural styles, WWW infrastructure



- This Week in REST <http://thisweekinrest.wordpress.com>
 - Bi-weekly blog on recent news about the REST style

Agenda

- Introduction
 - Motivation for modeling RESTful systems
- FSM Model of REST
 - REST introduction
 - eNFA introduction
 - Mapping REST to eNFA
 - Example Web application
- Closing Remarks
 - Future Work
 - Conclusion

63 slides?!?

Representational State Transfer (REST)

- Software architectural style
 - Abstract design principles
 - Distributed hypermedia systems
 - Scalability, simplicity, reliability ...
- Foundation of (a part of) the World Wide Web architecture
 - HTTP, URI, HTML



Importance of Understanding REST



- Understanding and evolving the WWW
 - The WWW has grown in scale and complexity
 - Where are we now and where should we go?

Real-Time
Web

- Applying REST to other domains
 - WWW is only one instance of REST
 - Can REST be applied to other domains? How?

WEB OF THINGS

- Engineering
 - *Understanding* is the basis for *doing* and *doing well*
 - Software frameworks, tools, ...



YAHOO! REST
discuss

Problems with Understanding REST

- Lack of simple and operational formal models
- Existing models
 - Semi-formal diagrams and natural language descriptions
 - Formal models of hypermedia systems (not REST)
 - Models focused on the WWW (not REST)
 - Separate client and server
 - Static, non-operational
 - Misuse of terminology



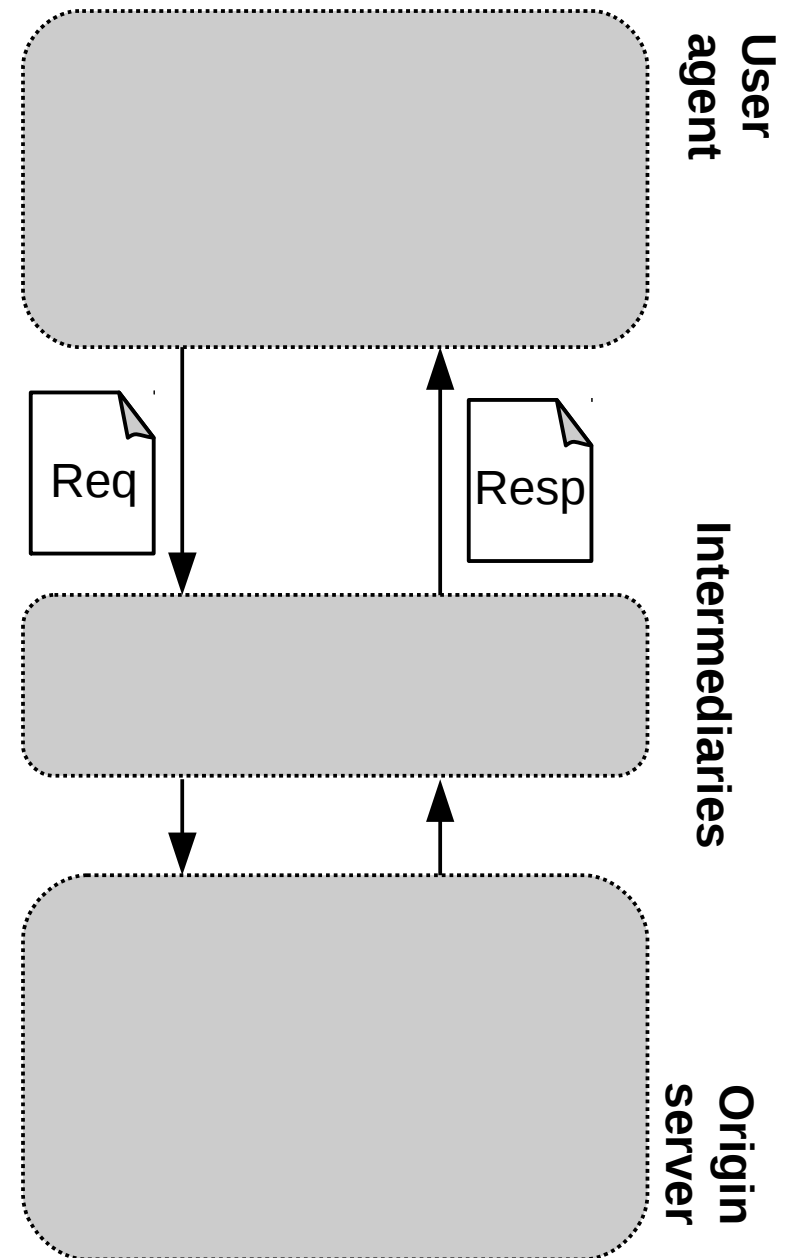
Research Goal – Formalism for Modeling RESTful Systems

- Any RESTful system
 - The WWW is a guide, not a judge
- System as a whole
 - Integrated view of both the client and server operation (the *application*)
- Operational
 - Both the static and dynamic view of operation
- Simple, understandable by researchers and engineers
 - “*Use the least powerful language suitable for expressing information, constraints or programs on the World Wide Web.*”, 2006, W3C TAG
- Use established concepts and terminology
 - Dr. Fielding's thesis

REST Principles 101

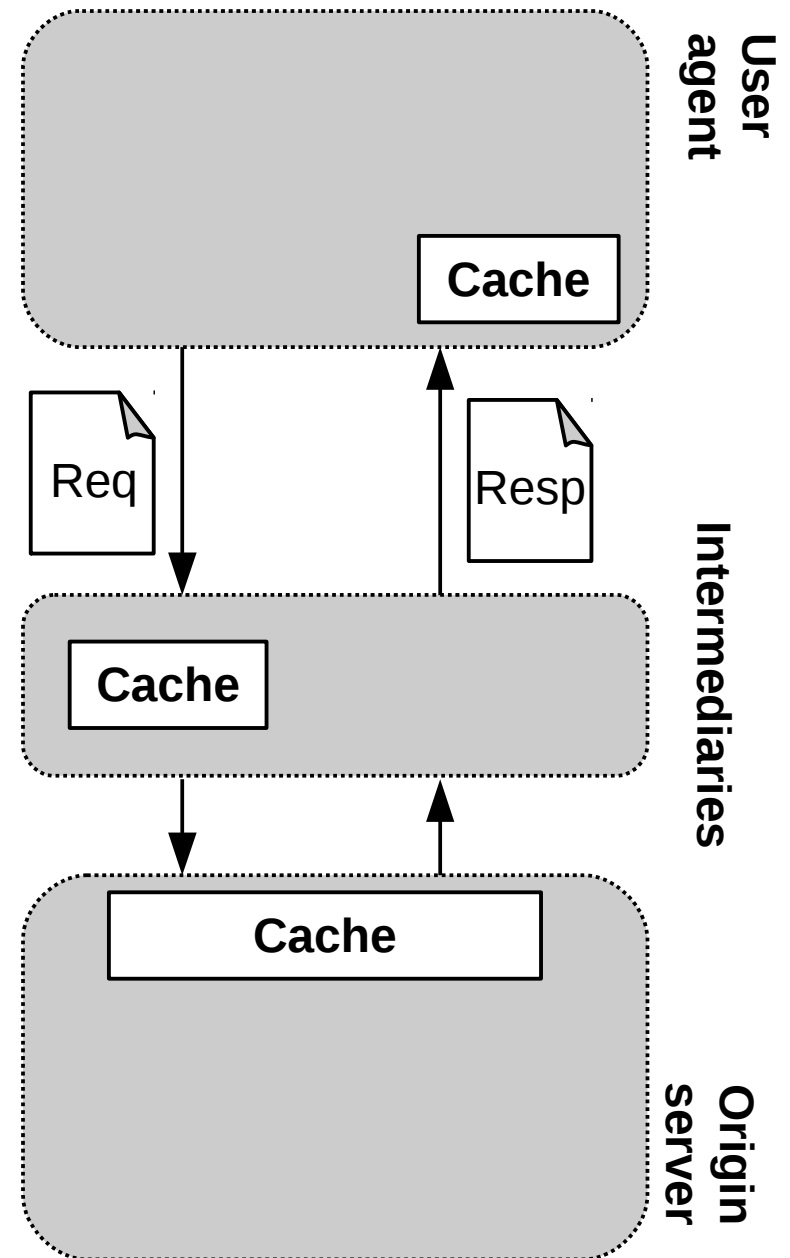
REST Principles 101

- Layered client-server



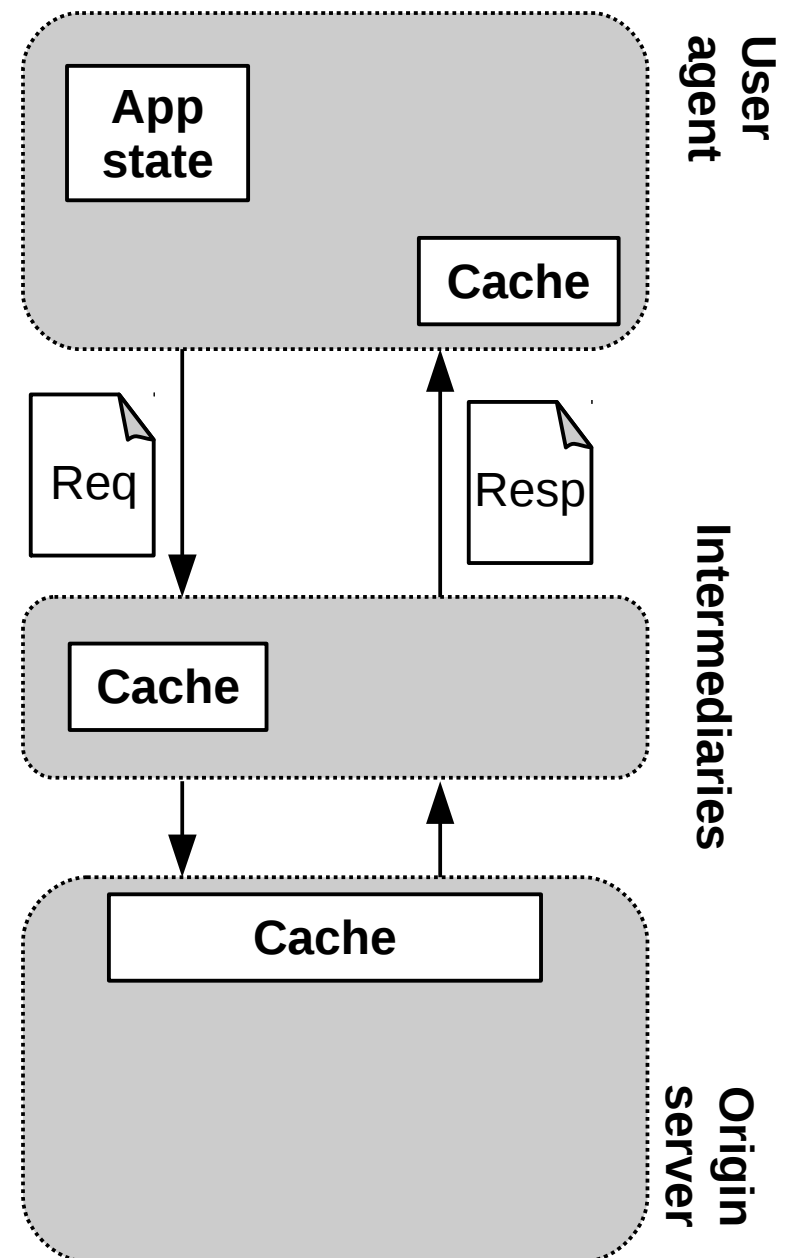
REST Principles 101

- Layered client-server
- Cacheable



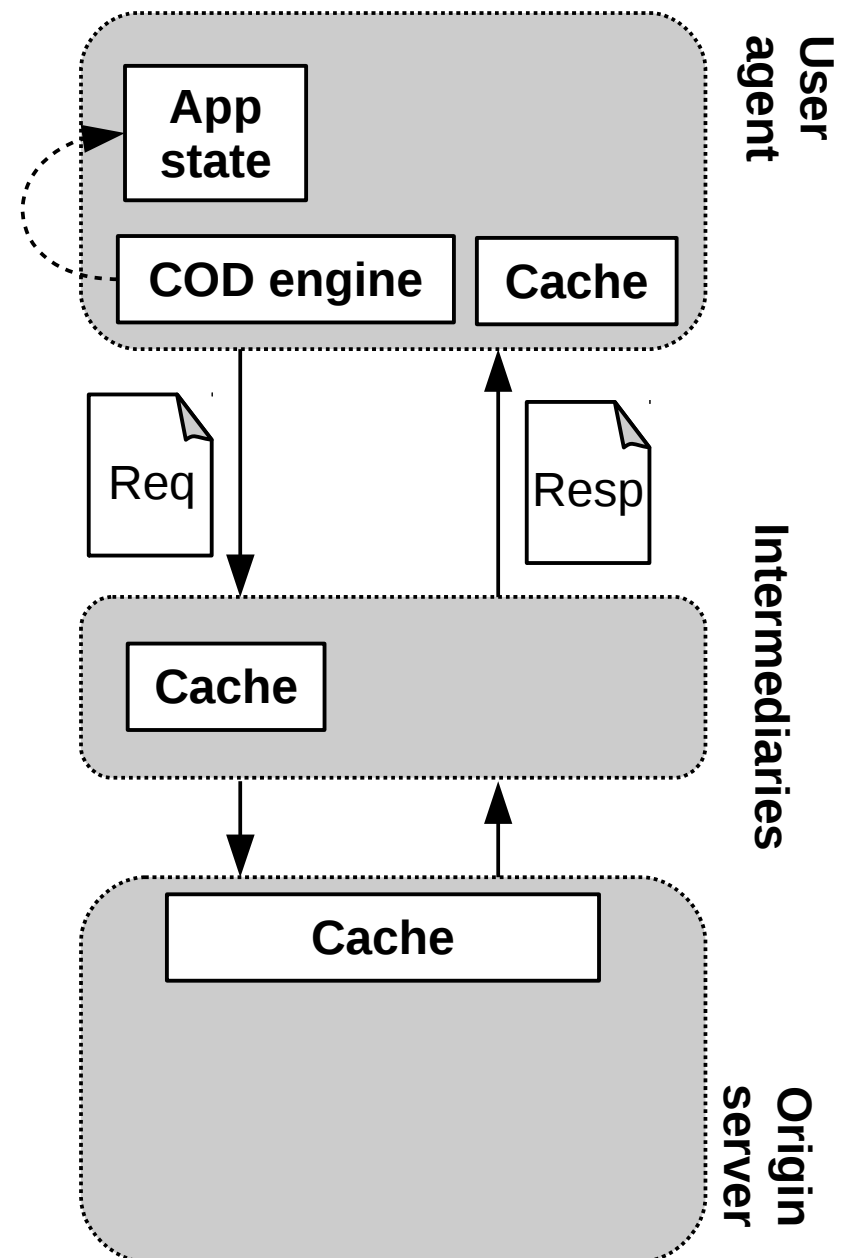
REST Principles 101

- Layered client-server
- Cacheable
- Stateless



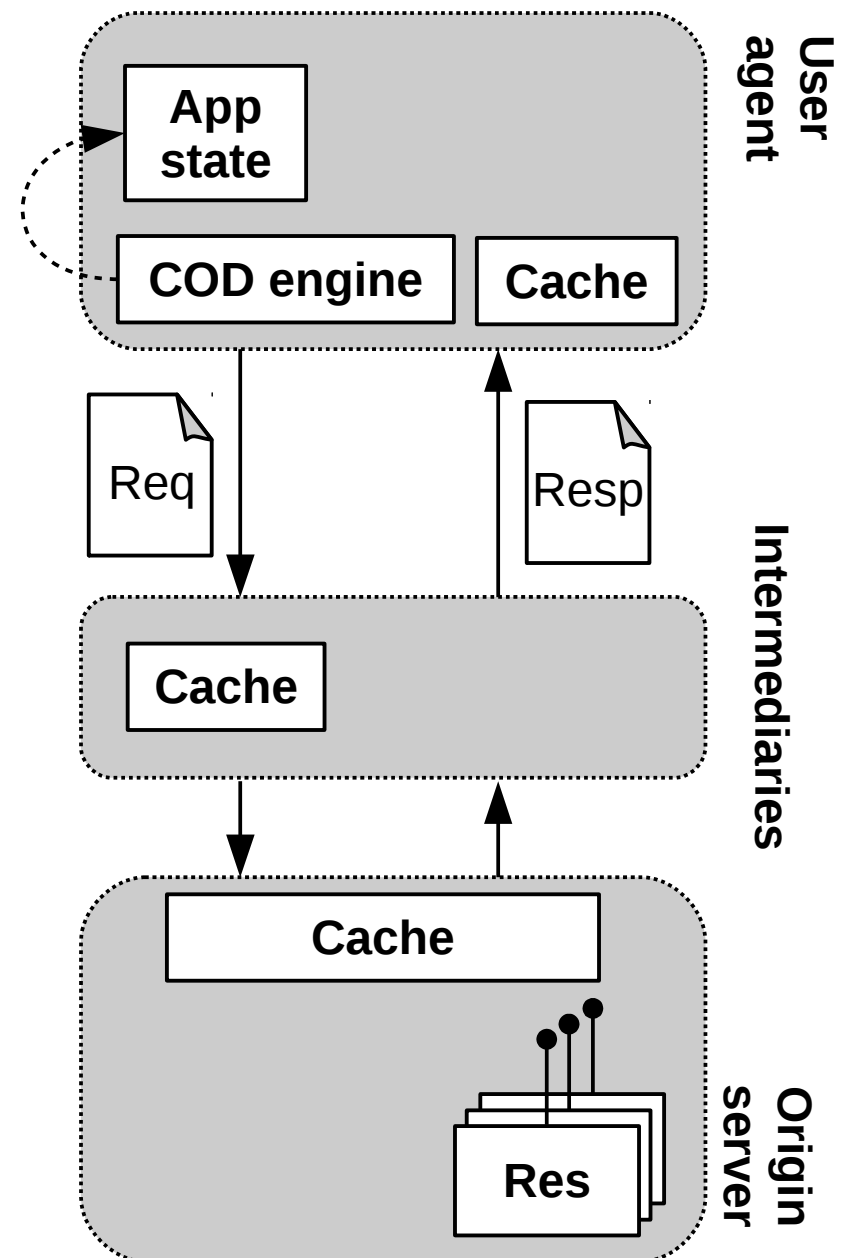
REST Principles 101

- Layered client-server
- Cacheable
- Stateless
- Code-on-demand



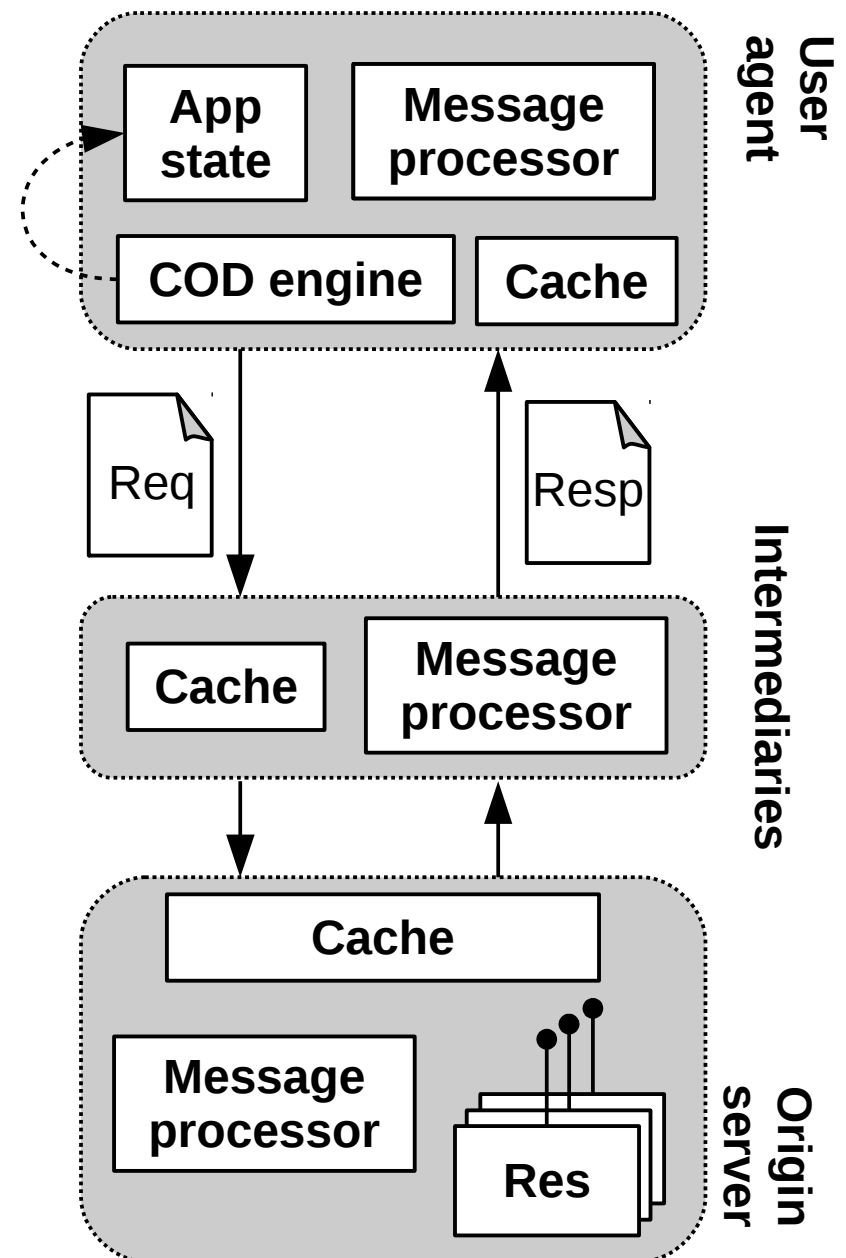
REST Principles 101

- Layered client-server
- Cacheable
- Stateless
- Code-on-demand
- Uniform interface
 - Identification of resources
 - Manipulation of resources through representations



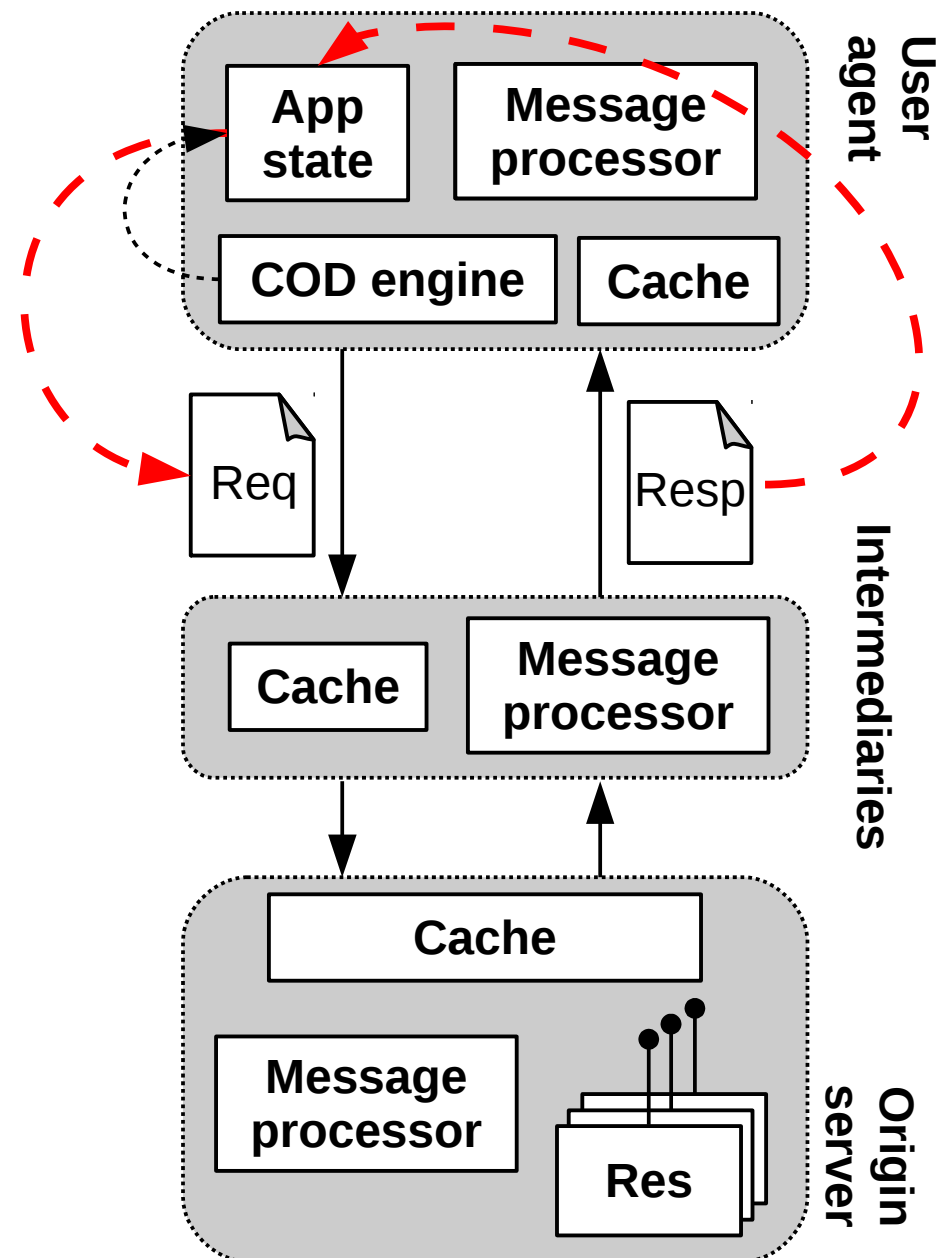
REST Principles 101

- Layered client-server
- Cacheable
- Stateless
- Code-on-demand
- Uniform interface
 - Identification of resources
 - Manipulation of resources through representations
 - Self-descriptive messages
 - Operations, media types, metadata ...



REST Principles 101

- Layered client-server
- Cacheable
- Stateless
- Code-on-demand
- Uniform interface
 - Identification of resources
 - Manipulation of resources through representations
 - Self-descriptive messages
 - Operations, media types, metadata ...
 - Hypermedia as the engine of application state
 - Links

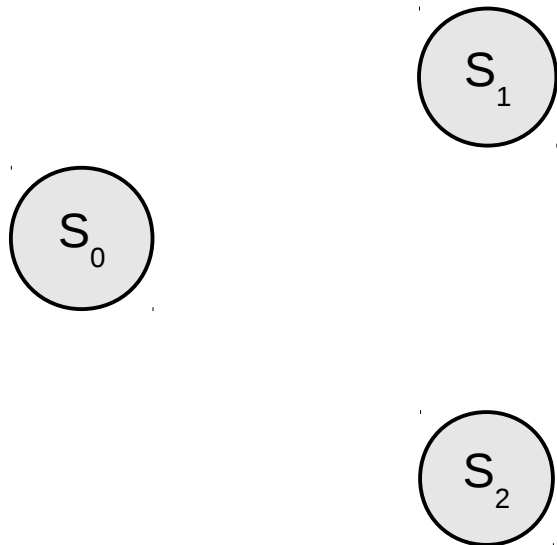


Finite State Machines

- ϵ -NFA formalism

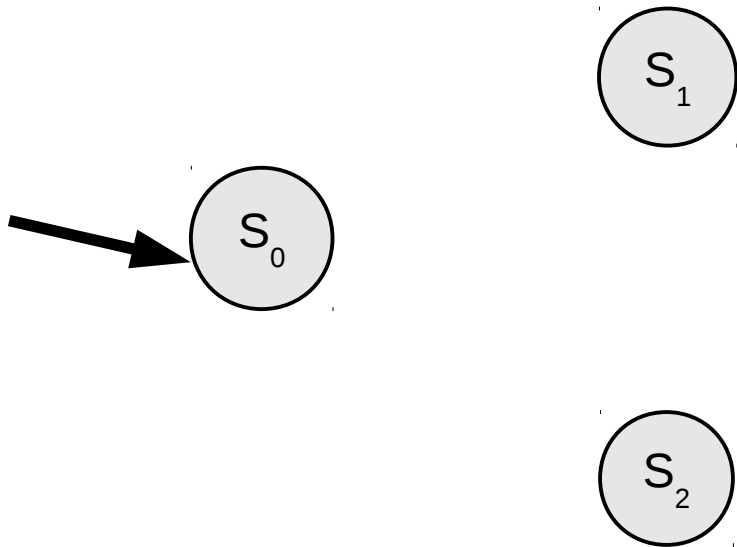
Finite State Machines

- ϵ -NFA formalism
 - Finite state machine



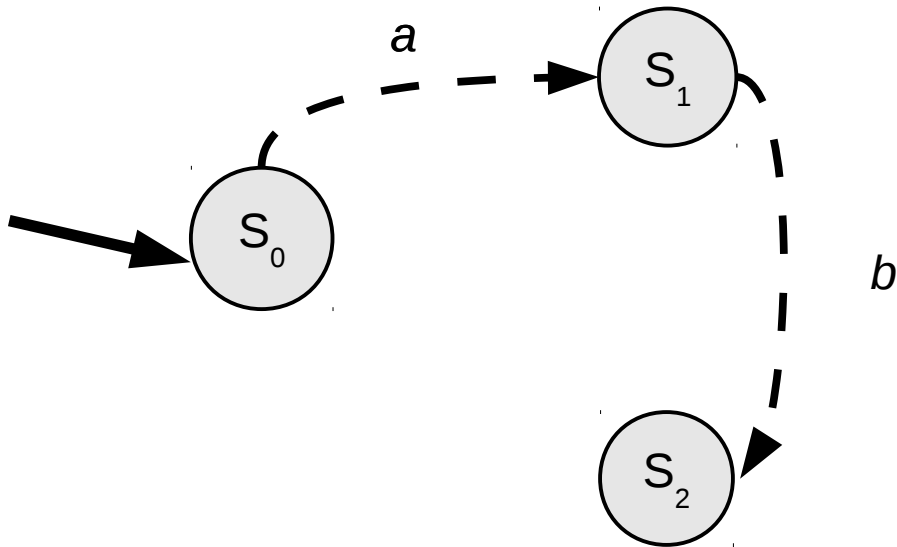
Finite State Machines

- ϵ -NFA formalism
 - Finite state machine



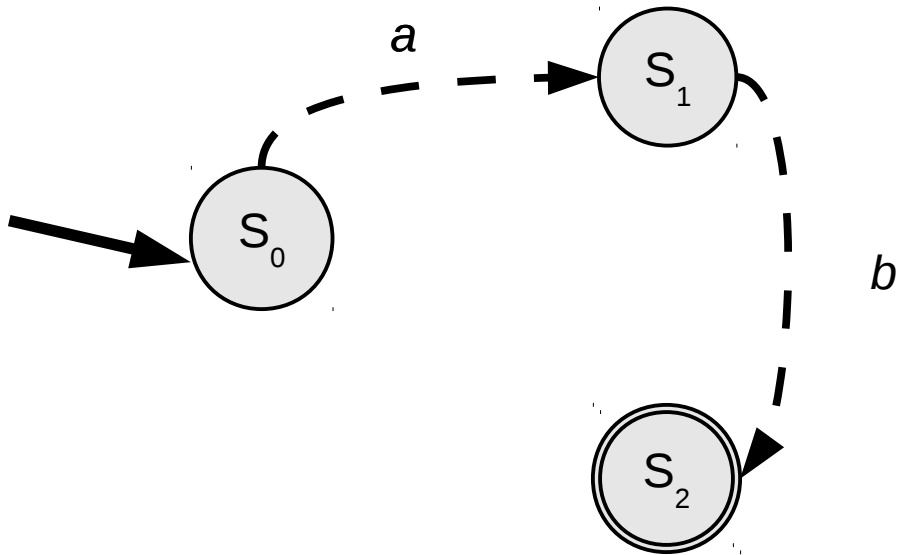
Finite State Machines

- ϵ -NFA formalism
 - Finite state machine



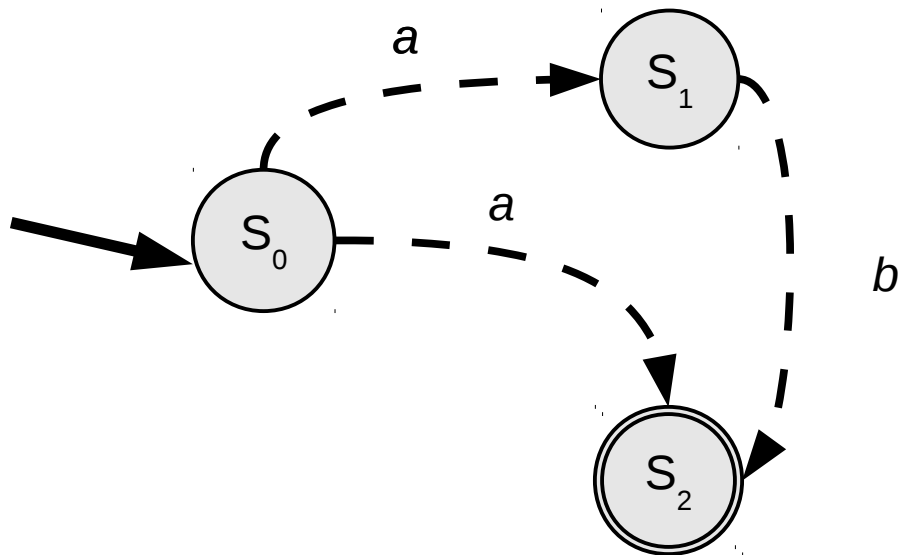
Finite State Machines

- ϵ -NFA formalism
 - Finite state machine



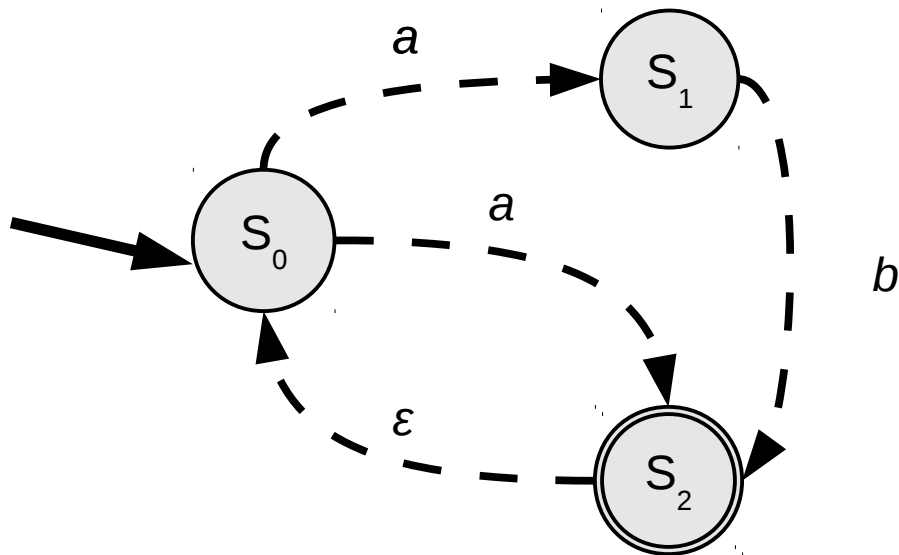
Finite State Machines

- ϵ -NFA formalism
 - Finite state machine
 - Nondeterministic



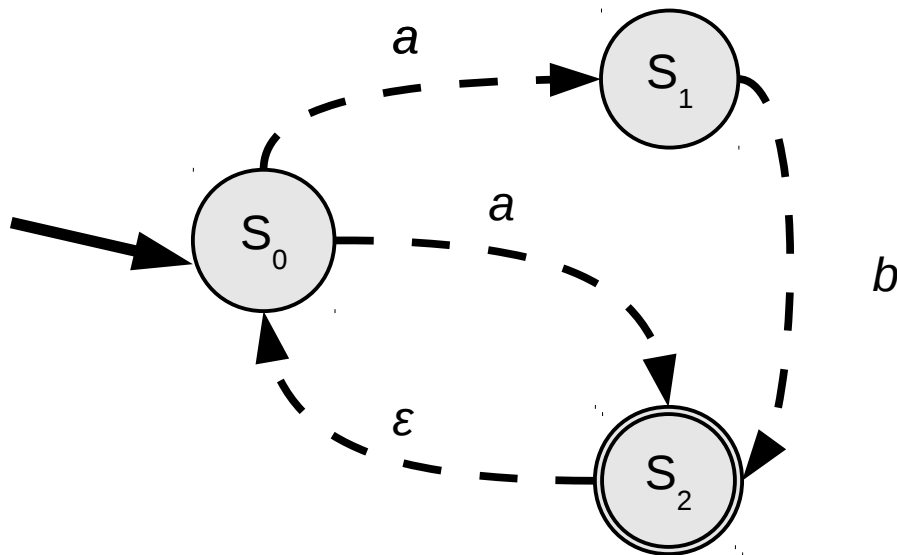
Finite State Machines

- ϵ -NFA formalism
 - Finite state machine
 - Nondeterministic
 - Epsilon transitions



Finite State Machines

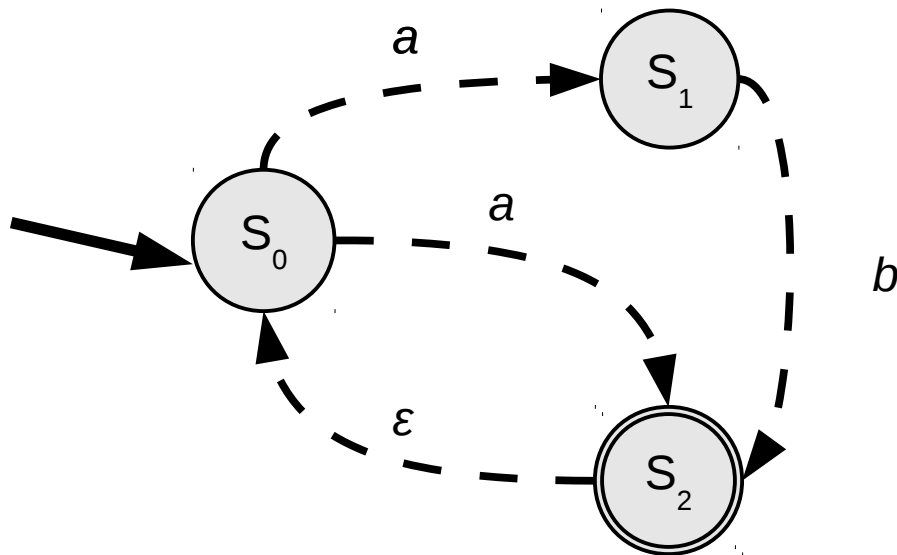
- ϵ -NFA formalism
 - Finite state machine
 - Nondeterministic
 - Epsilon transitions
 - *Transition Function:*
 $States \times (Inputs \cup \epsilon) \rightarrow P(States)$



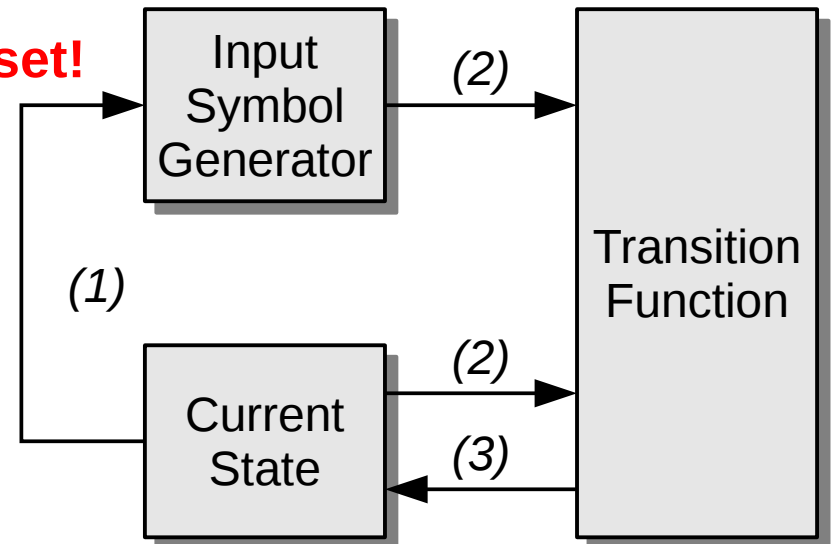
Power set!

Finite State Machines

- ϵ -NFA formalism
 - Finite state machine
 - Nondeterministic
 - Epsilon transitions
 - *Transition Function:*
 $States \times (Inputs \cup \epsilon) \rightarrow P(States)$
- System level view



Power set!



Mapping REST to eNFA

- FSM = (States, Inputs, Initial, TransitionFunction, Accepting)

Mapping REST to eNFA

- FSM = (States, Inputs, Initial, TransitionFunction, Accepting)
 - **States = AppStates, AppStates $\subseteq P(\text{Representations}) - \{\}$**
 - Application state is a subset of all possible representations of all resources

Mapping REST to eNFA

- FSM = (States, Inputs, Initial, TransitionFunction, Accepting)
 - **States = AppStates, AppStates $\subseteq P(\text{Representations}) - \{\}$**
 - Application state is a subset of all possible representations of all resources
 - **Initial = initial application state at system startup**
 - Representation with links to entry-points of known applications

Mapping REST to eNFA

- FSM = (States, Inputs, Initial, TransitionFunction, Accepting)
 - **States = AppStates, AppStates $\subseteq P(\text{Representations}) - \{\}$**
 - Application state is a subset of all possible representations of all resources
 - **Initial = initial application state at system startup**
 - Representation with links to entry-points of known applications
 - **Inputs $\subseteq (\text{Requests} \times \text{LinkTypes})$**
 - Requests $\subseteq (\text{Operations} \times \text{ResourceIDs} \times \text{Representations})$

Importance of Link Types

- Different state transition semantics

```
GET /photo.jpg HTTP/1.1
```

```
HTTP/1.1 200 OK
```



- Navigation to image (<a>) or embedding in a Web page ()?

Mapping REST to eNFA

- FSM = (States, Inputs, Initial, TransitionFunction, Accepting)
 - **States = AppStates, AppStates $\subseteq P(\text{Representations}) - \{\}$**
 - Application state is a subset of all possible representations of all resources
 - **Initial = initial application state at system startup**
 - Representation with links to entry-points of known applications
 - **Inputs $\subseteq (\text{Requests} \times \text{LinkTypes})$**
 - Requests $\subseteq (\text{Operations} \times \text{ResourceIDs} \times \text{Representations})$
 - **TransitionFunction : AppStates $\times ((\text{Requests} \times \text{LinkTypes}) \cup \epsilon) \rightarrow P(\text{AppStates})$**
 - Translation of input symbols into server requests
 - Processing of requests into responses (nondeterministic!)
 - Integration of response representations into the next application state
 - Code-on-demand transitions on the client

Mapping REST to eNFA

- FSM = (States, Inputs, Initial, TransitionFunction, Accepting)
 - **States = AppStates, AppStates $\subseteq P(\text{Representations}) - \{\}$**
 - Application state is a subset of all possible representations of all resources
 - **Initial = initial application state at system startup**
 - Representation with links to entry-points of known applications
 - **Inputs $\subseteq (\text{Requests} \times \text{LinkTypes})$**
 - Requests $\subseteq (\text{Operations} \times \text{ResourceIDs} \times \text{Representations})$
 - **TransitionFunction : AppStates $\times ((\text{Requests} \times \text{LinkTypes}) \cup \epsilon) \rightarrow P(\text{AppStates})$**
 - Translation of input symbols into server requests
 - Processing of requests into responses (nondeterministic!)
 - Integration of response representations into the next application state
 - Code-on-demand transitions on the client
 - **Accepting = SteadyStates, SteadyStates $\subseteq \text{AppStates}$**
 - In steady states, representations of all embedded resources are present in the application state

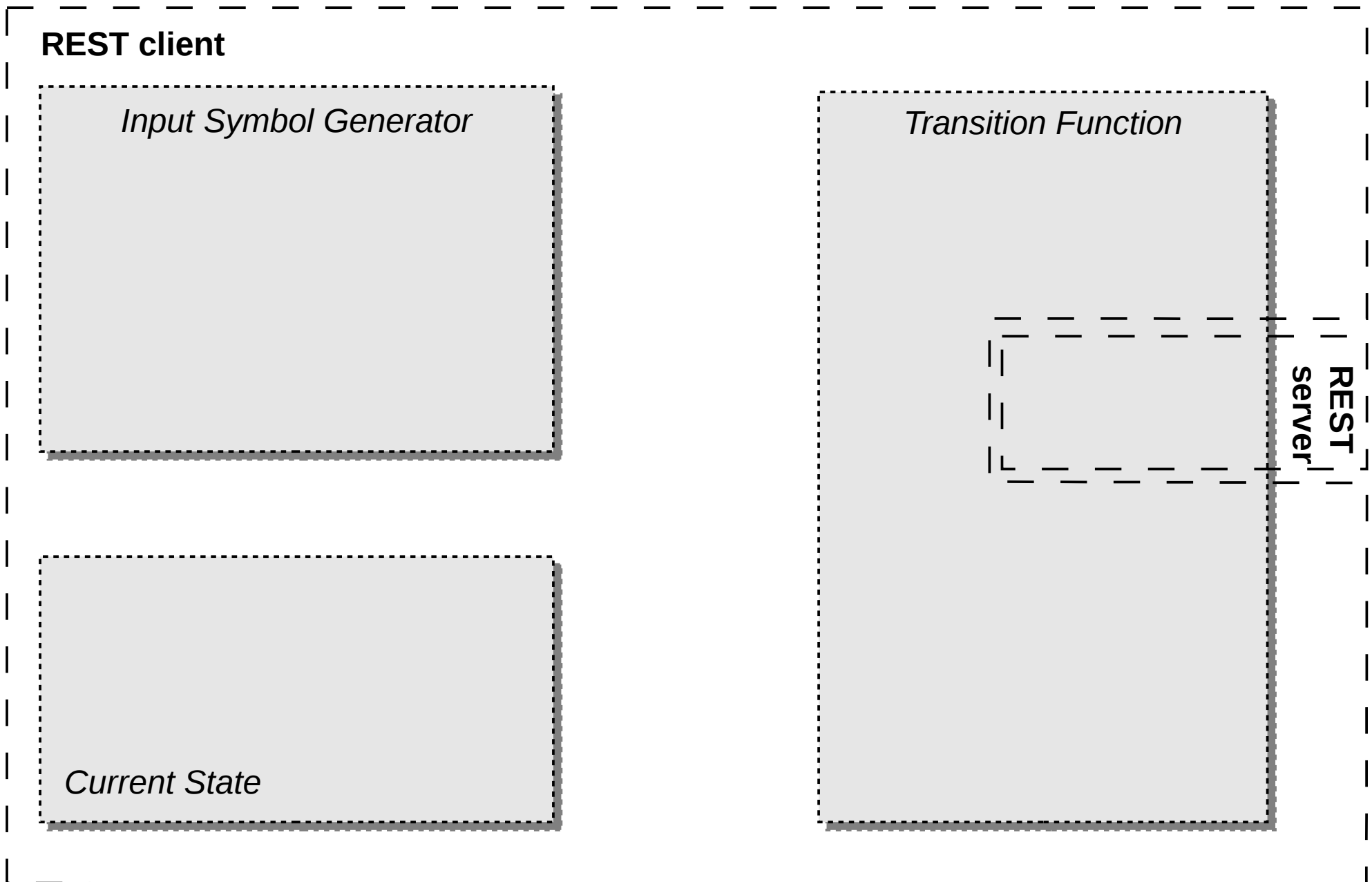
Mapping REST to eNFA

Input Symbol Generator

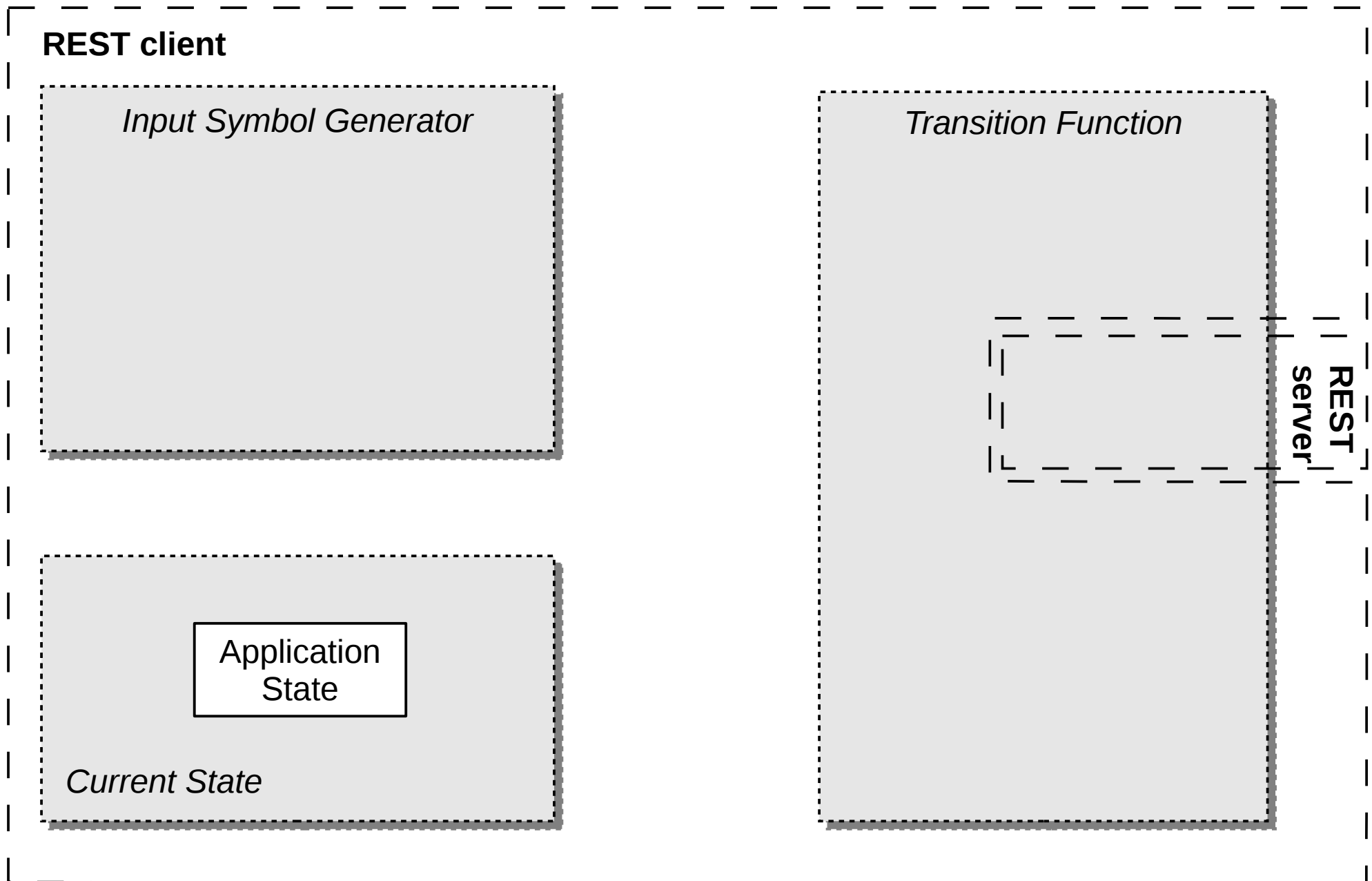
Transition Function

Current State

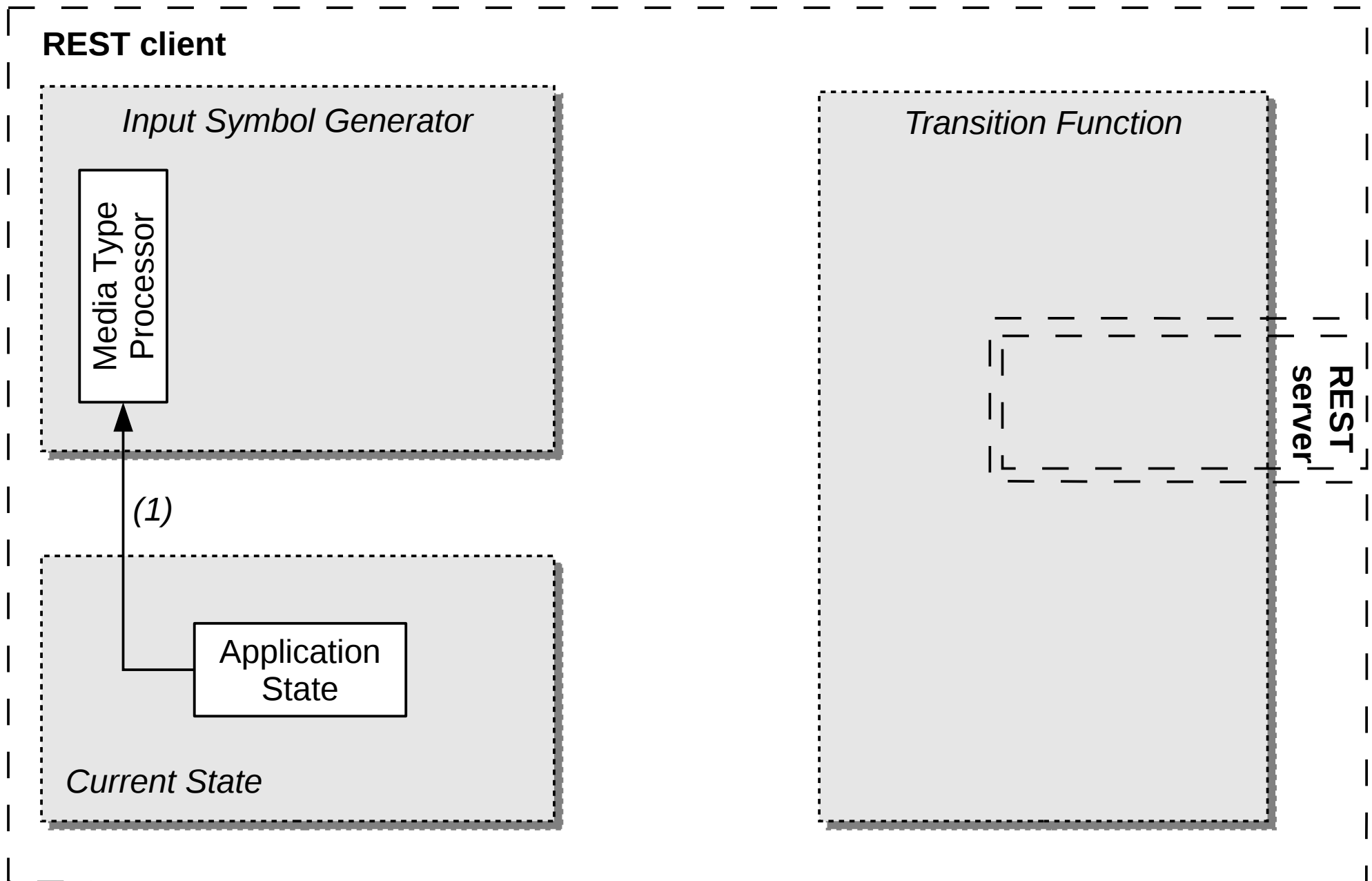
Mapping REST to eNFA



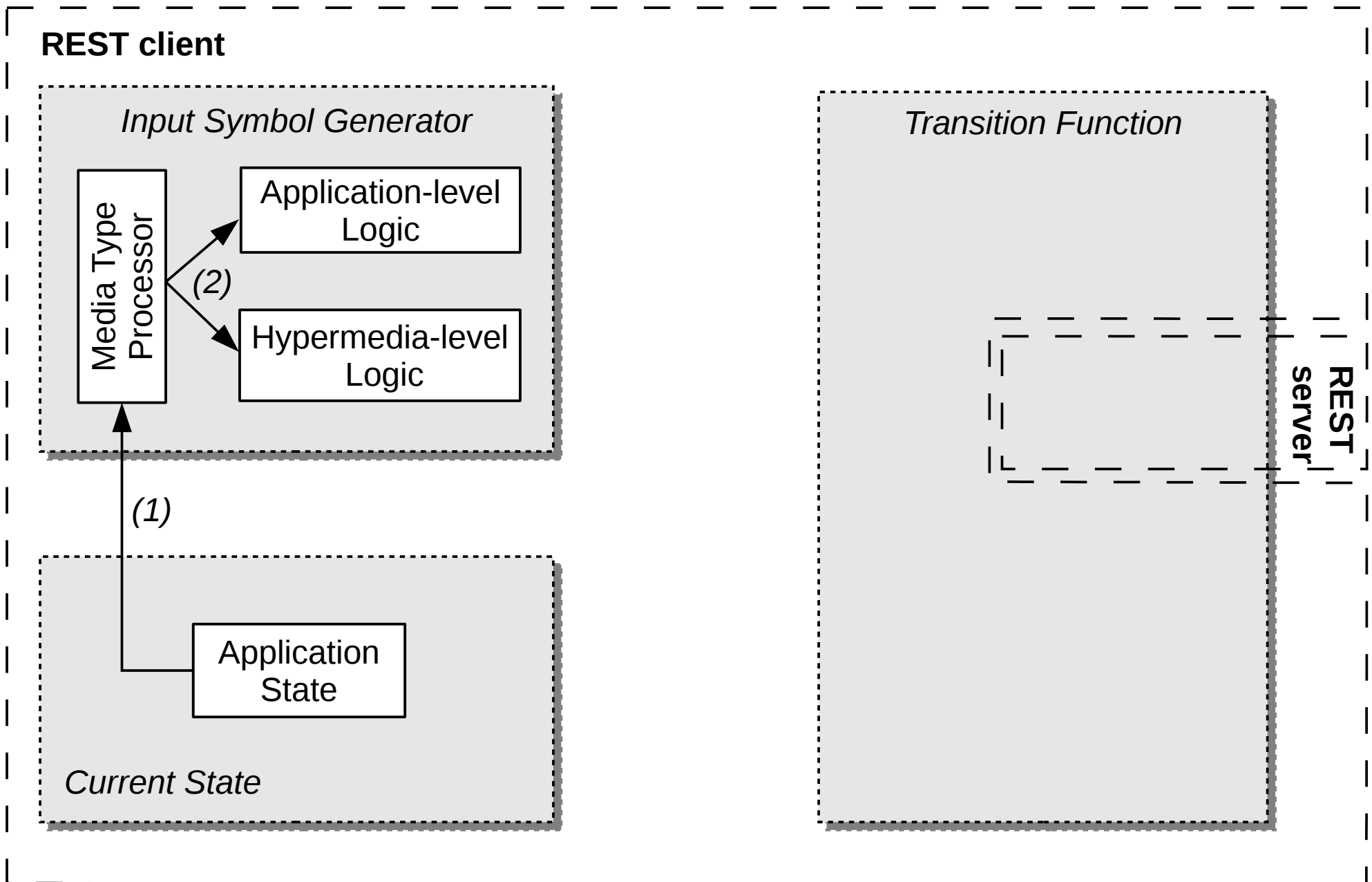
Mapping REST to eNFA



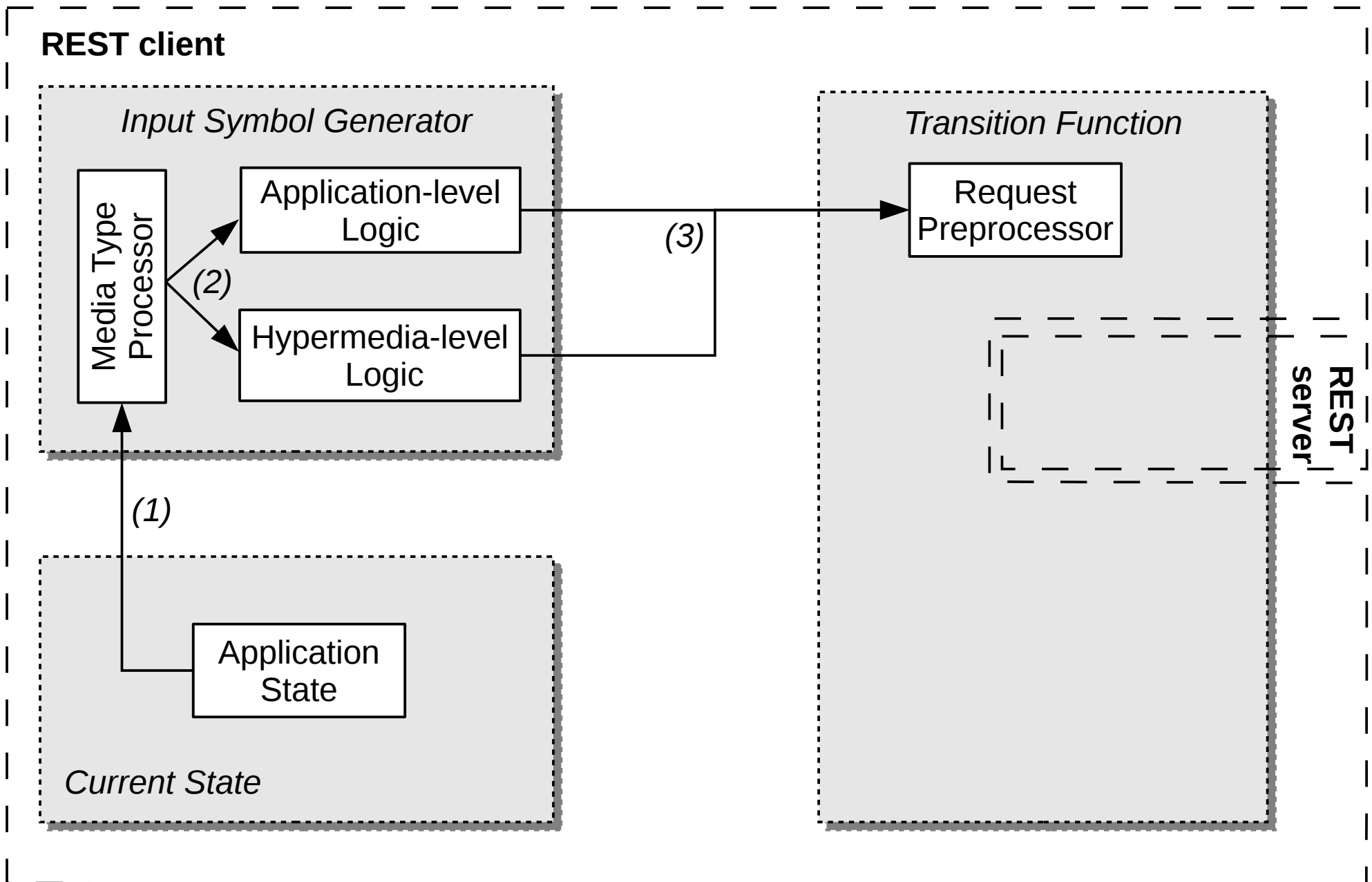
Mapping REST to eNFA



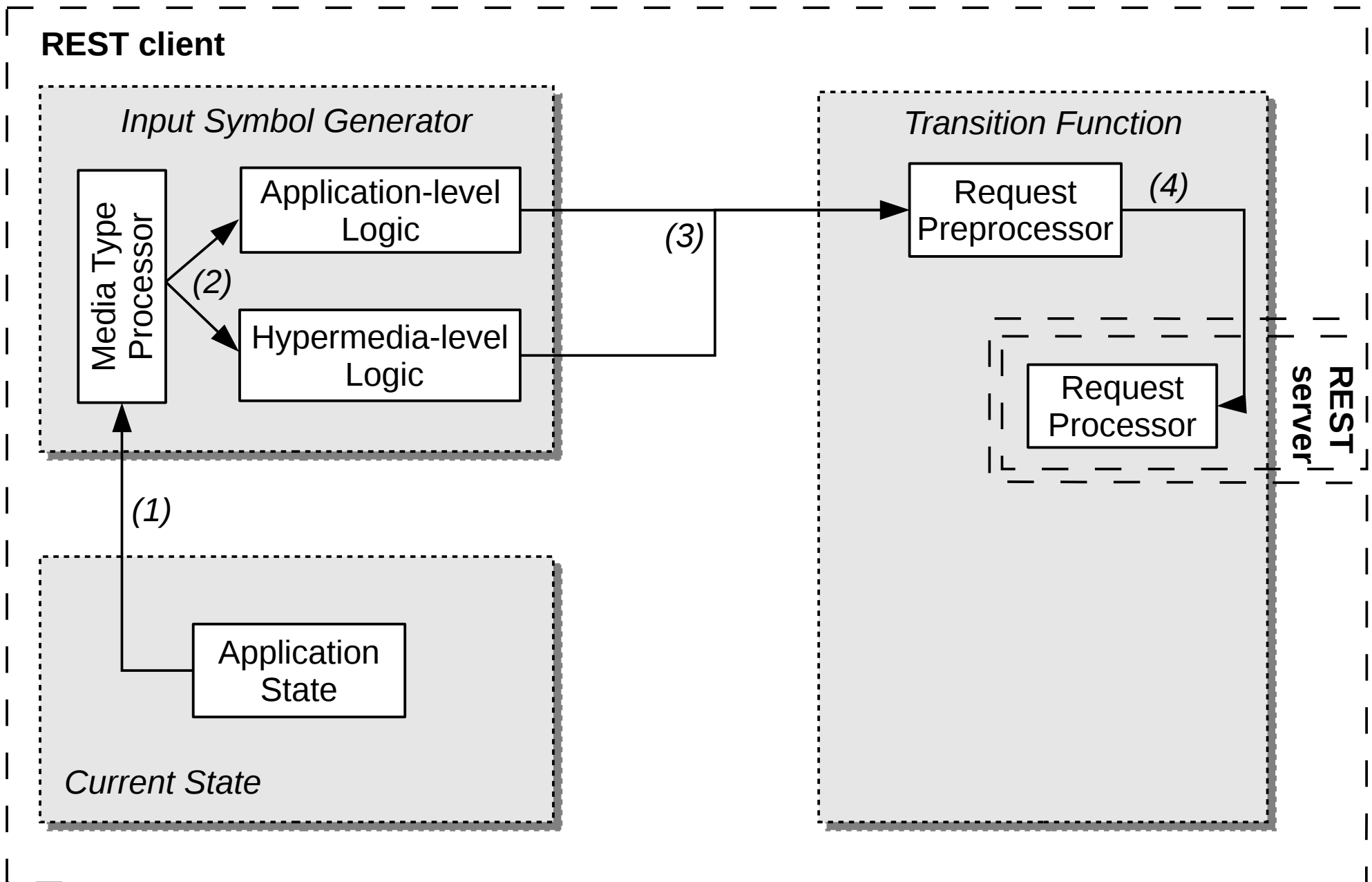
Mapping REST to eNFA



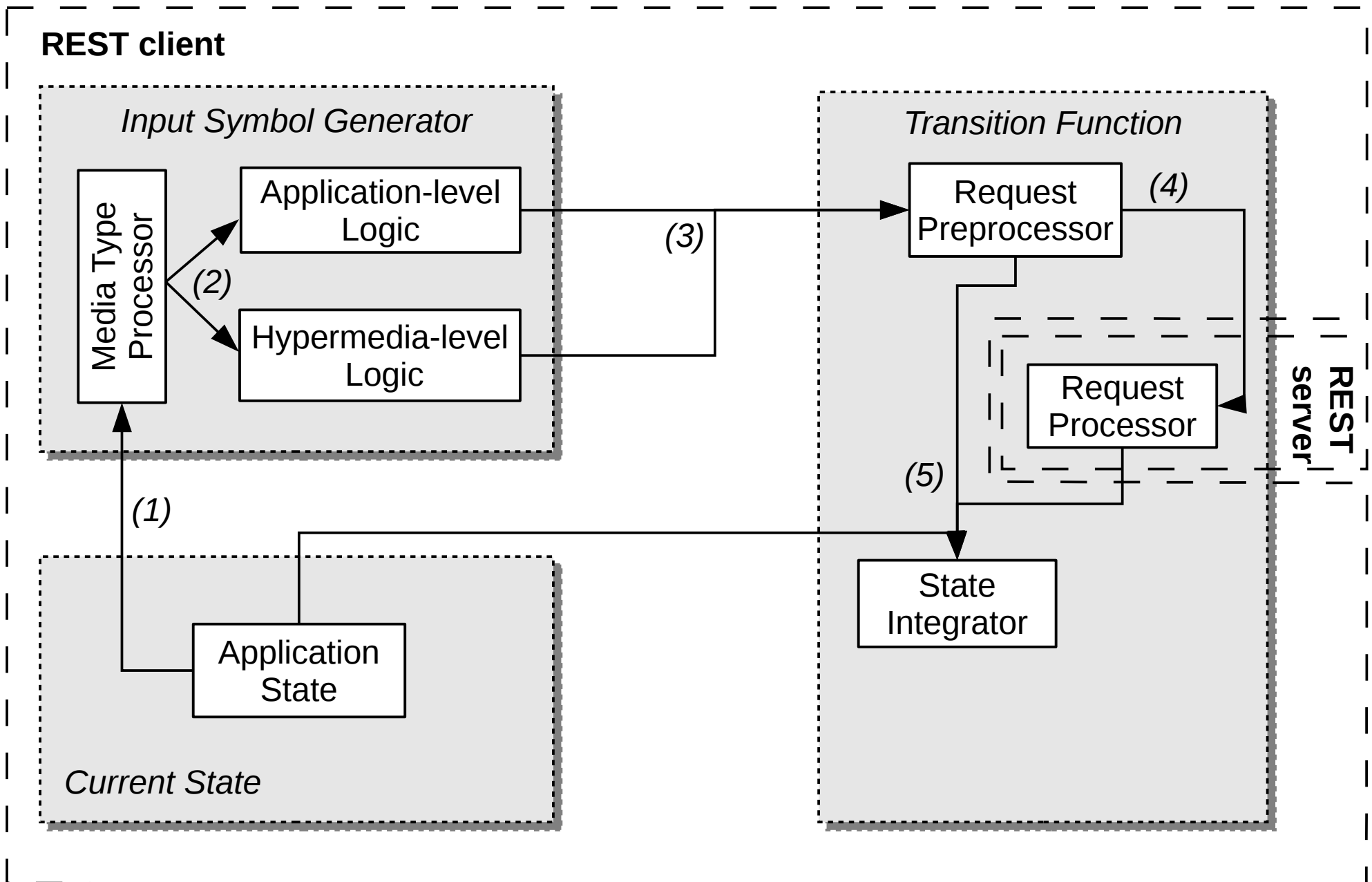
Mapping REST to eNFA



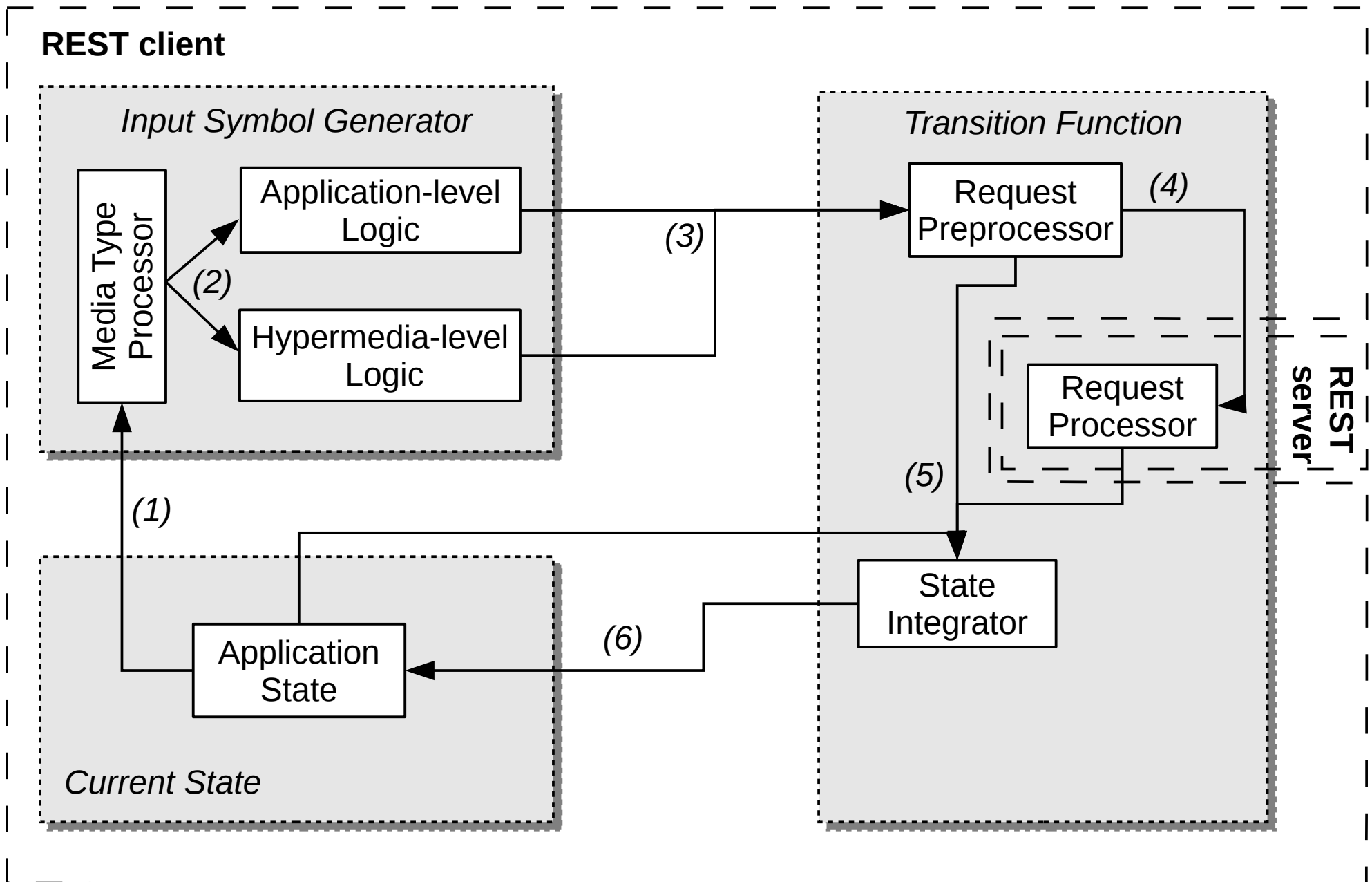
Mapping REST to eNFA



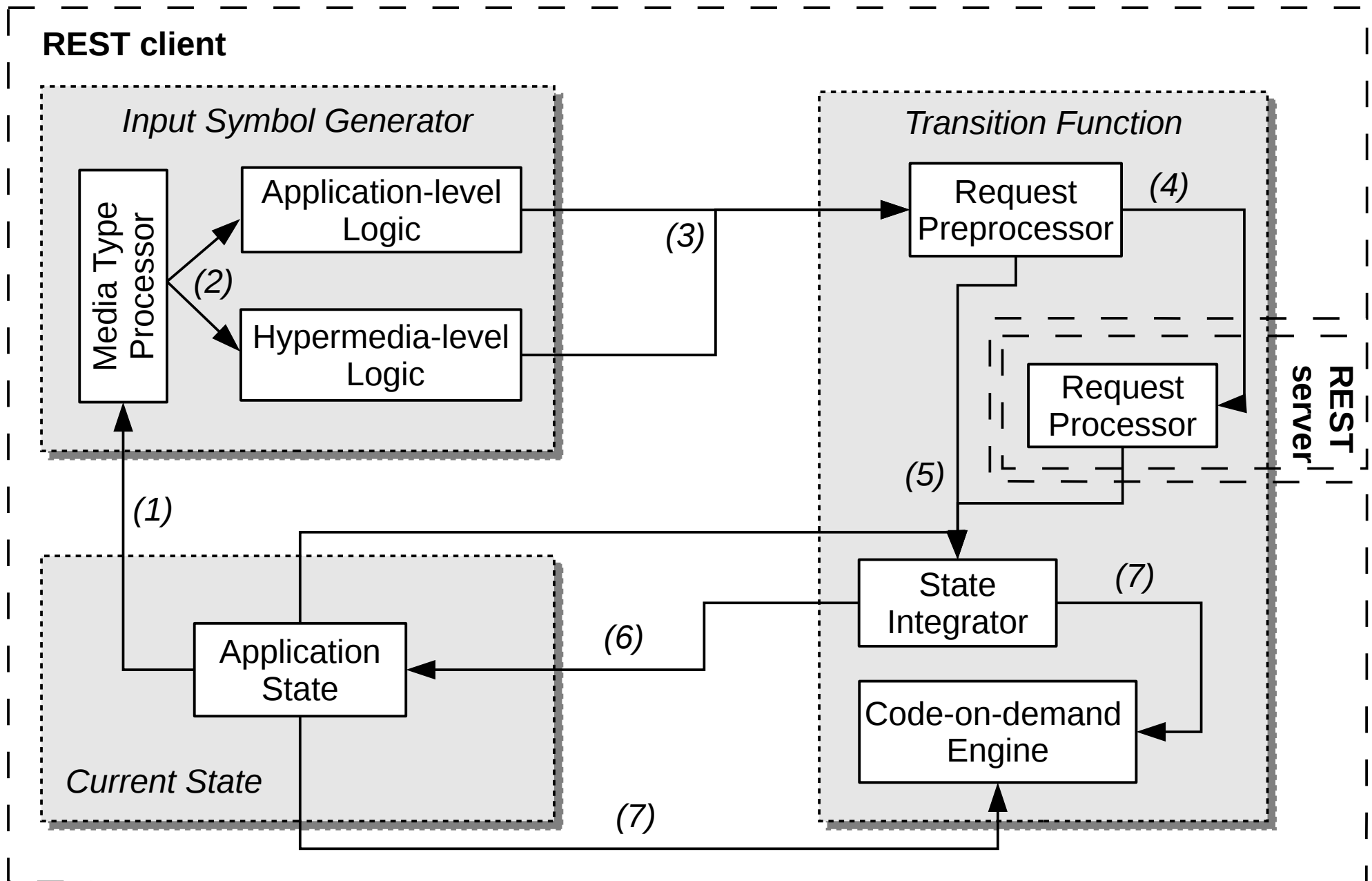
Mapping REST to eNFA



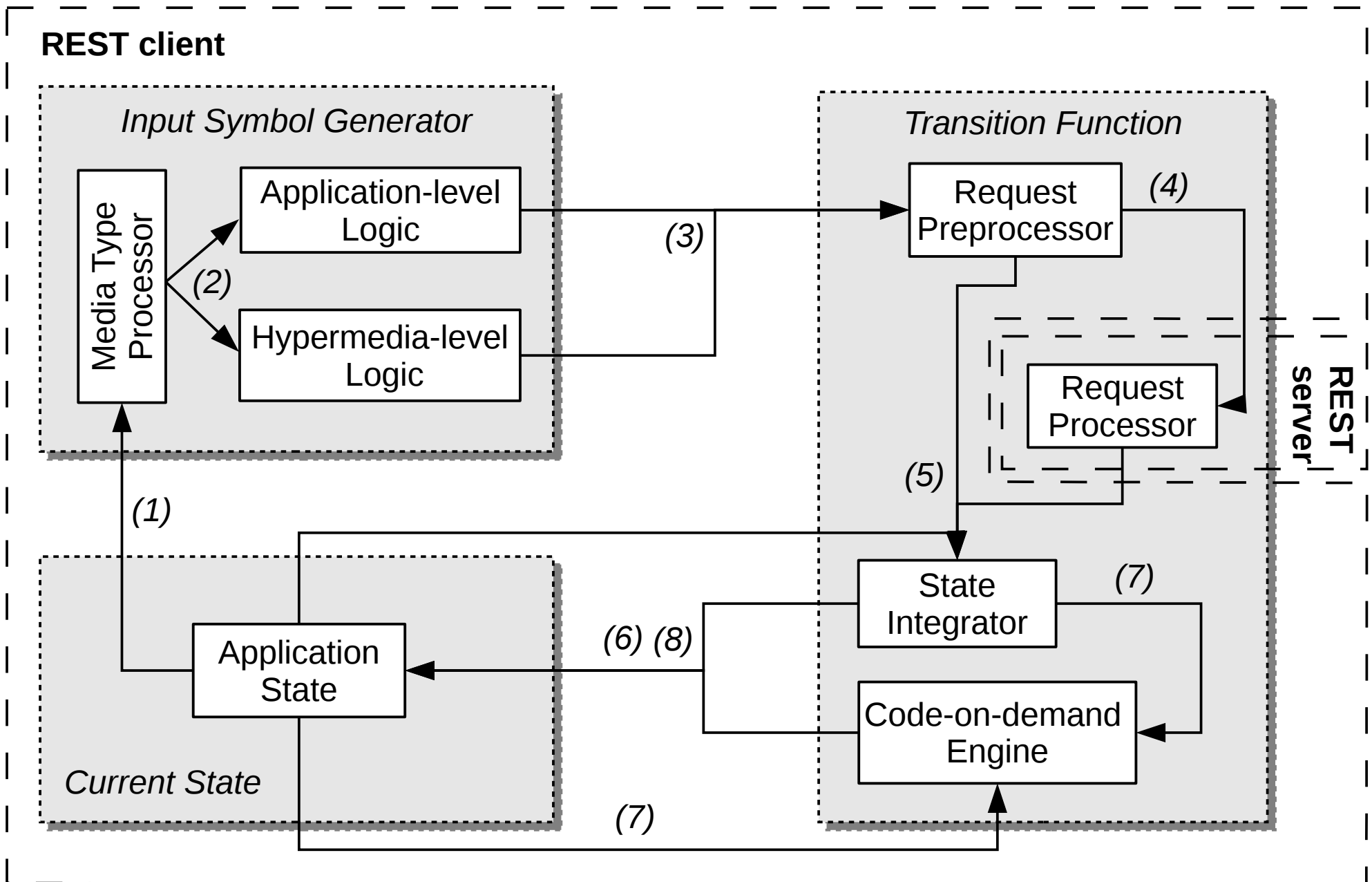
Mapping REST to eNFA



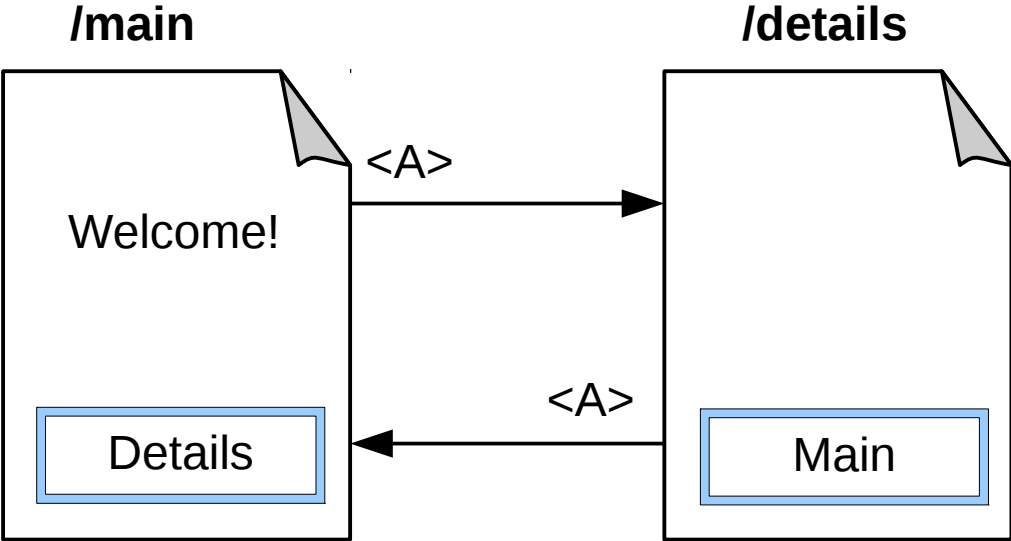
Mapping REST to eNFA



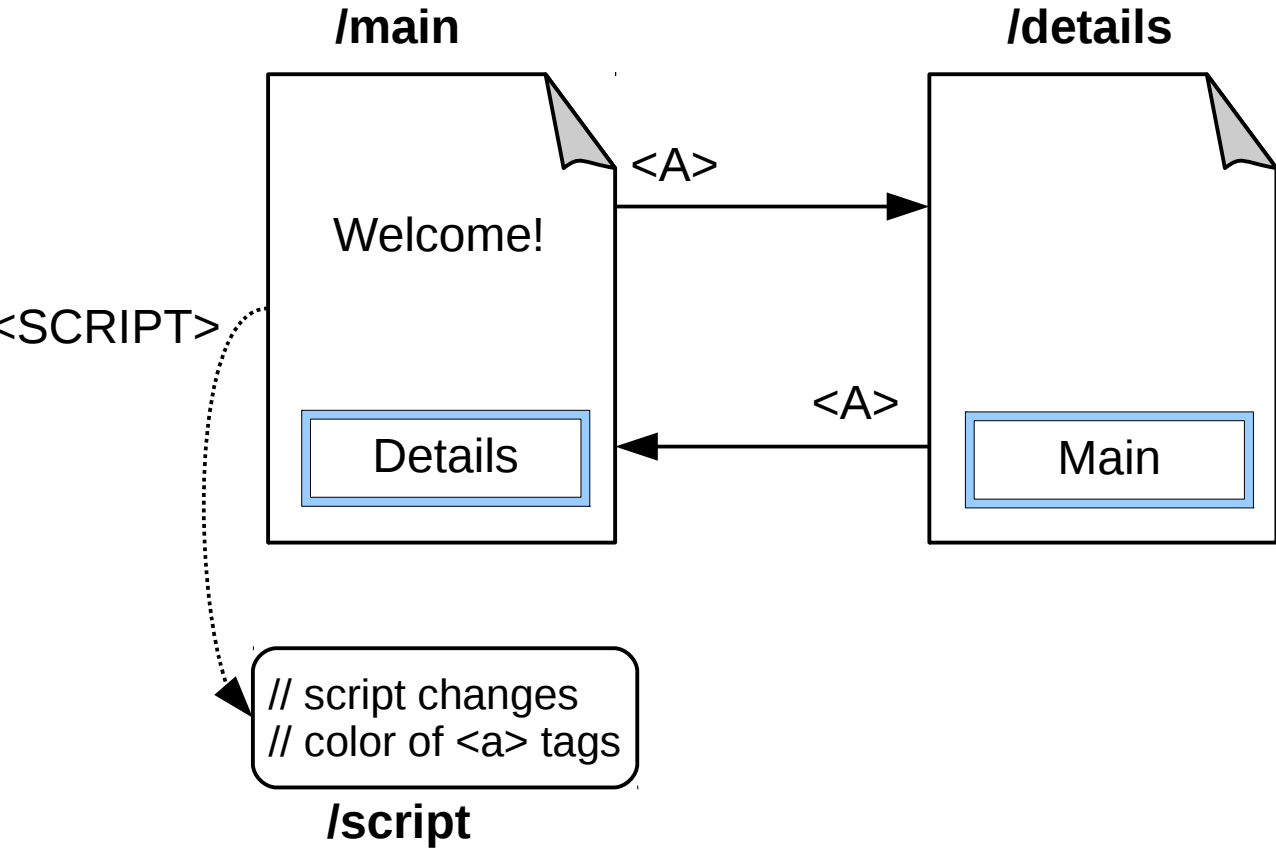
Mapping REST to eNFA



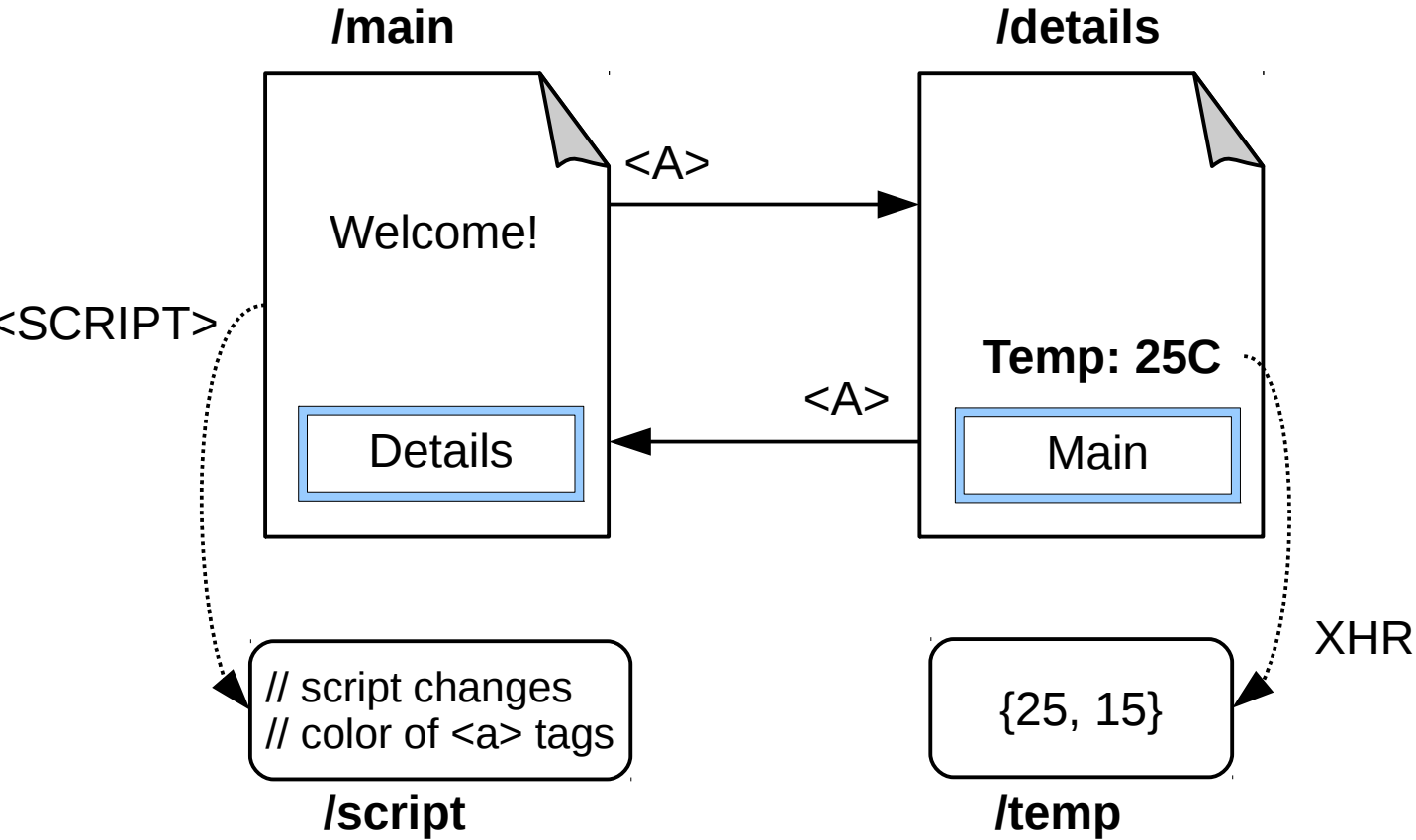
Example: Weather Forecast Web Application



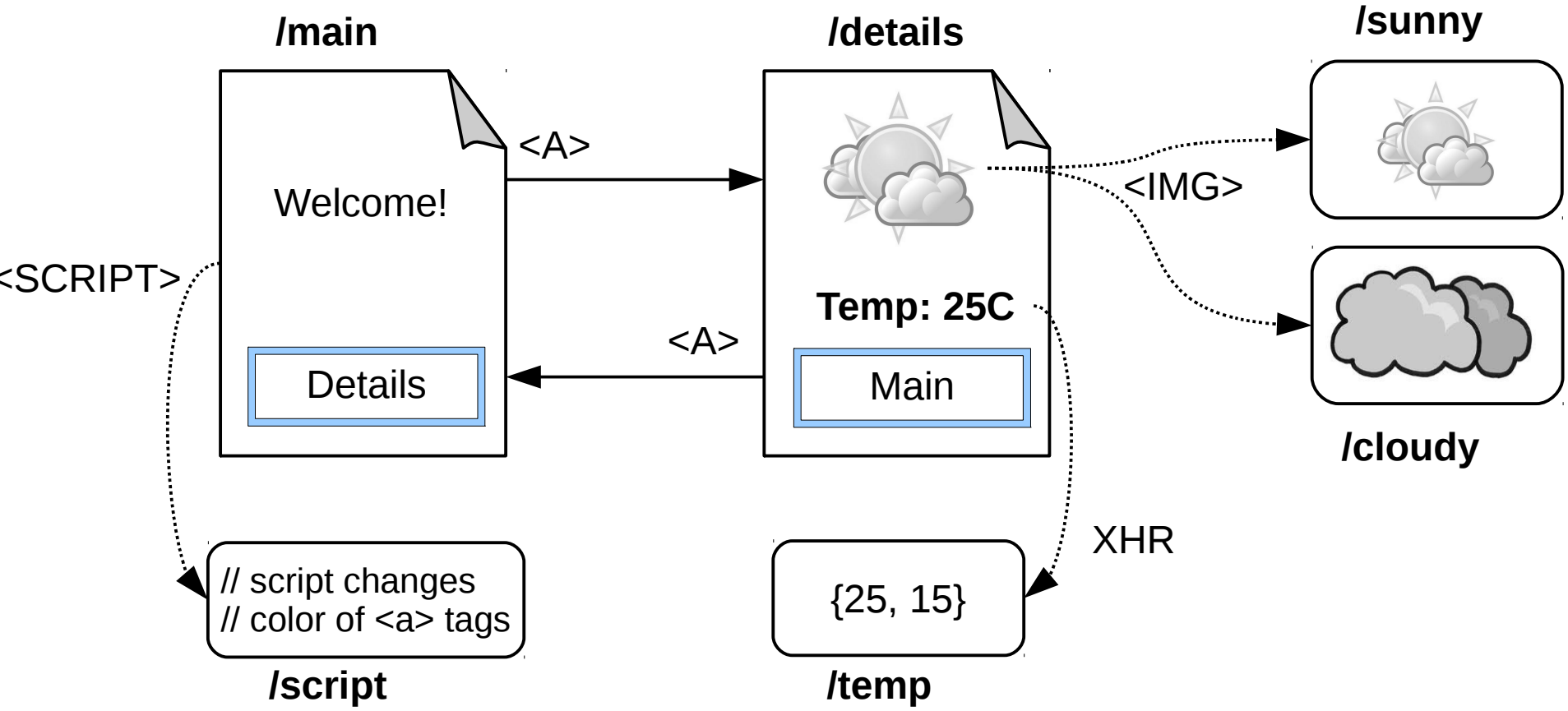
Example: Weather Forecast Web Application



Example: Weather Forecast Web Application

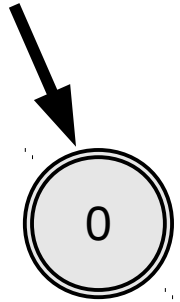


Example: Weather Forecast Web Application



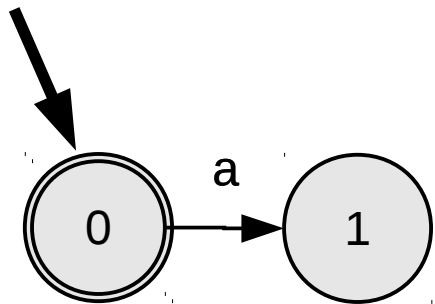

Example: Weather Forecast Web Application

Initial State



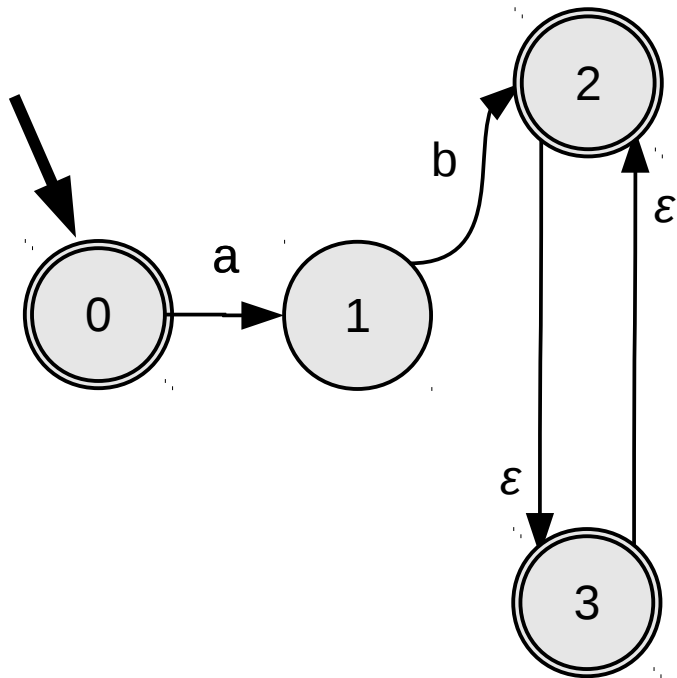
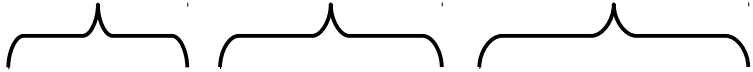
Example: Weather Forecast Web Application

Initial State Main page



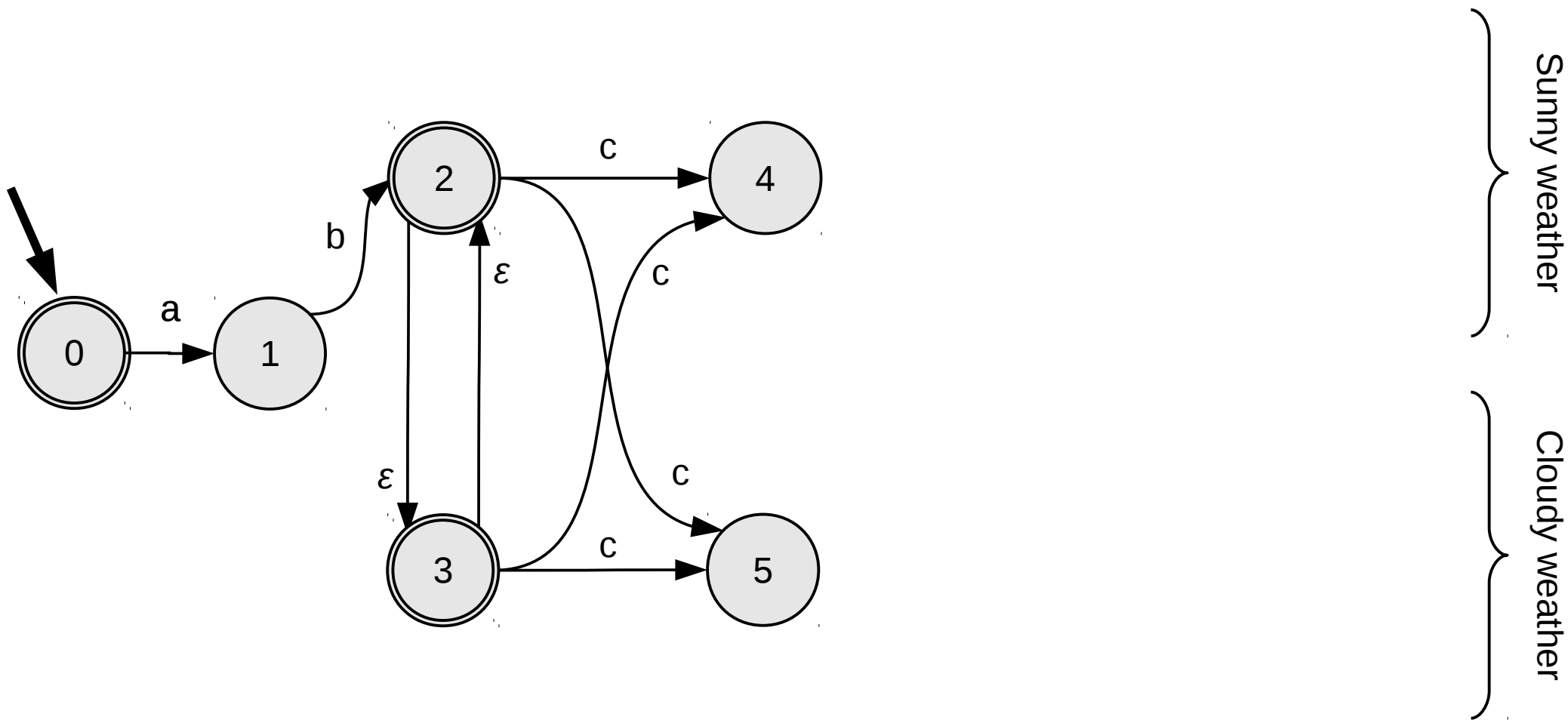
Example: Weather Forecast Web Application

Initial State Main page Script fetch and color changes



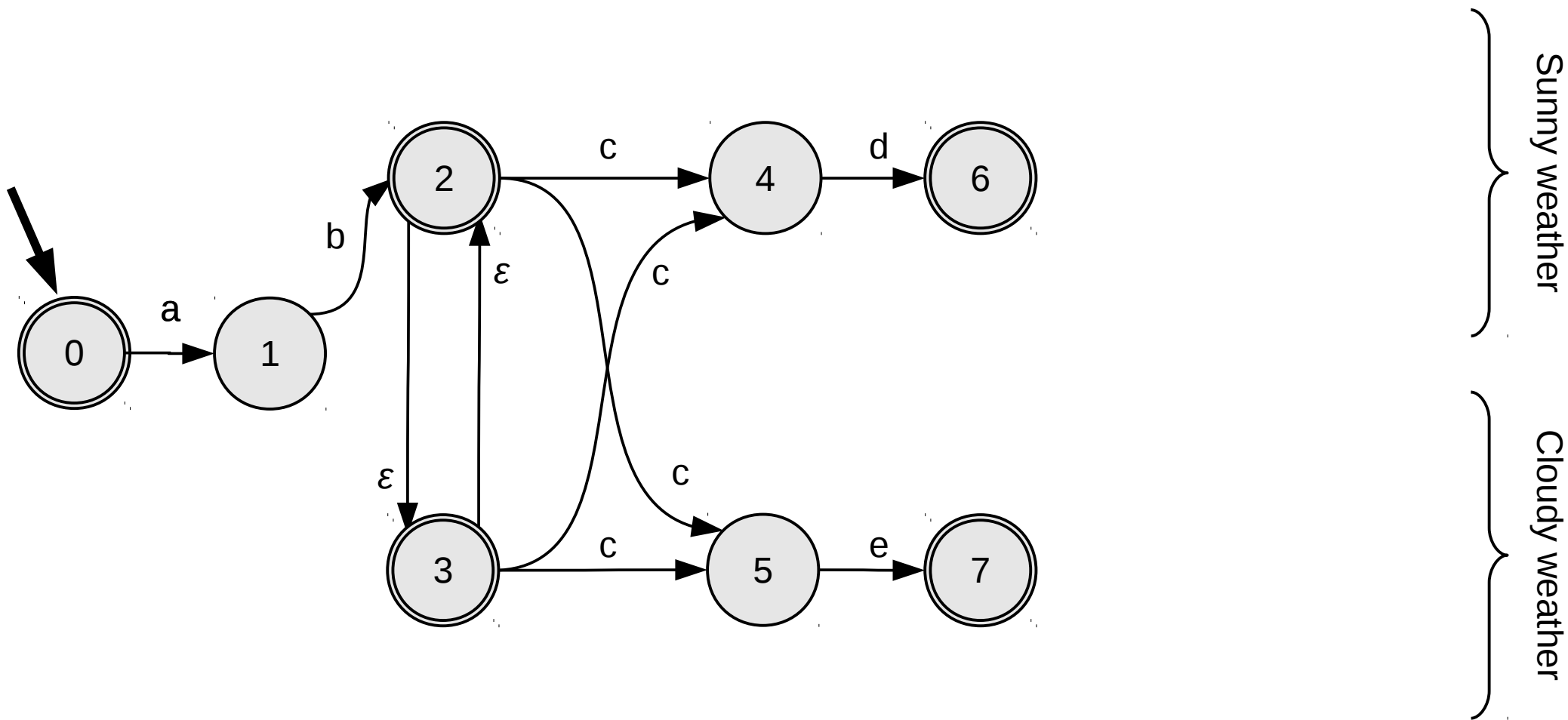
Example: Weather Forecast Web Application

Initial State Main page Script fetch and color changes Details page

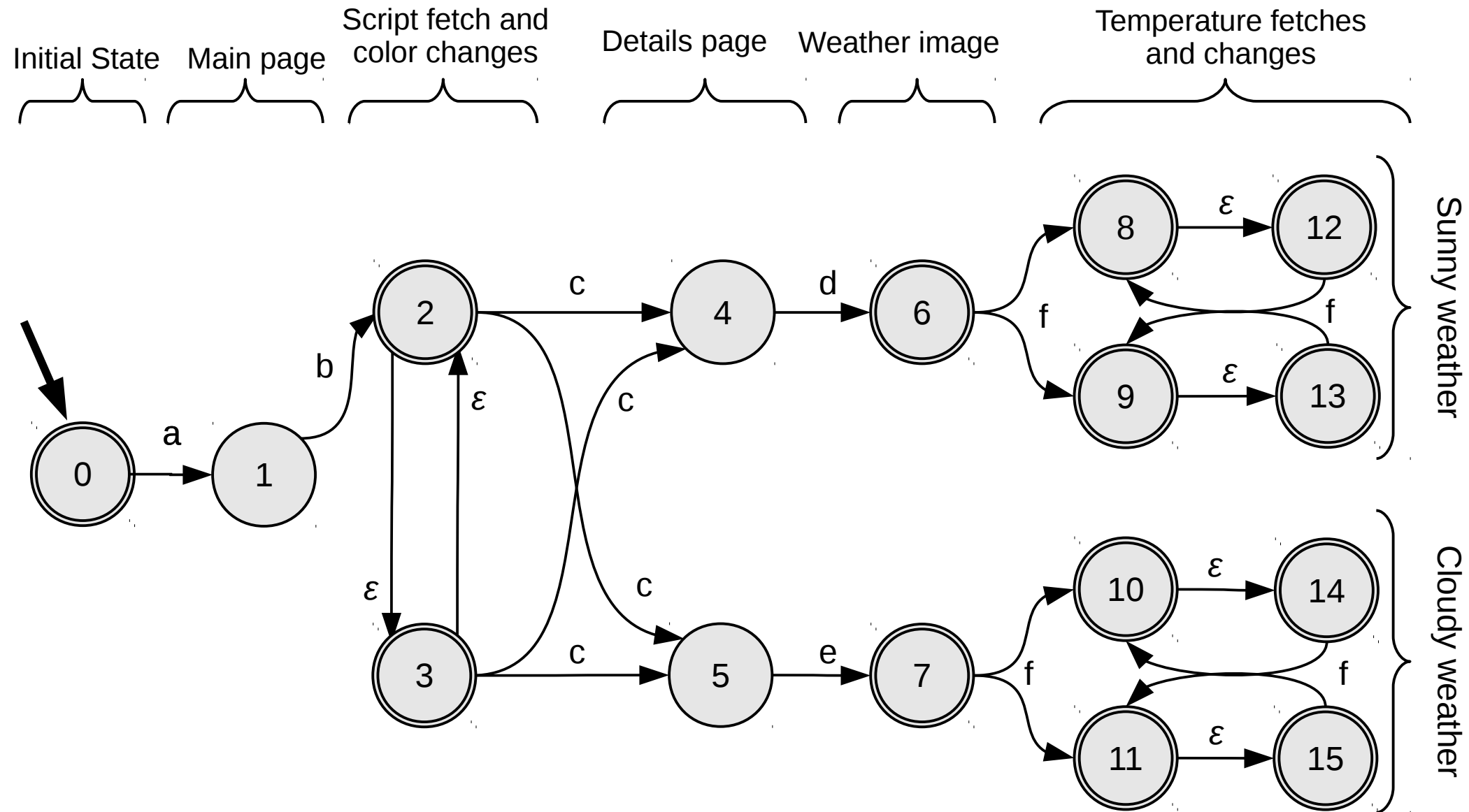


Example: Weather Forecast Web Application

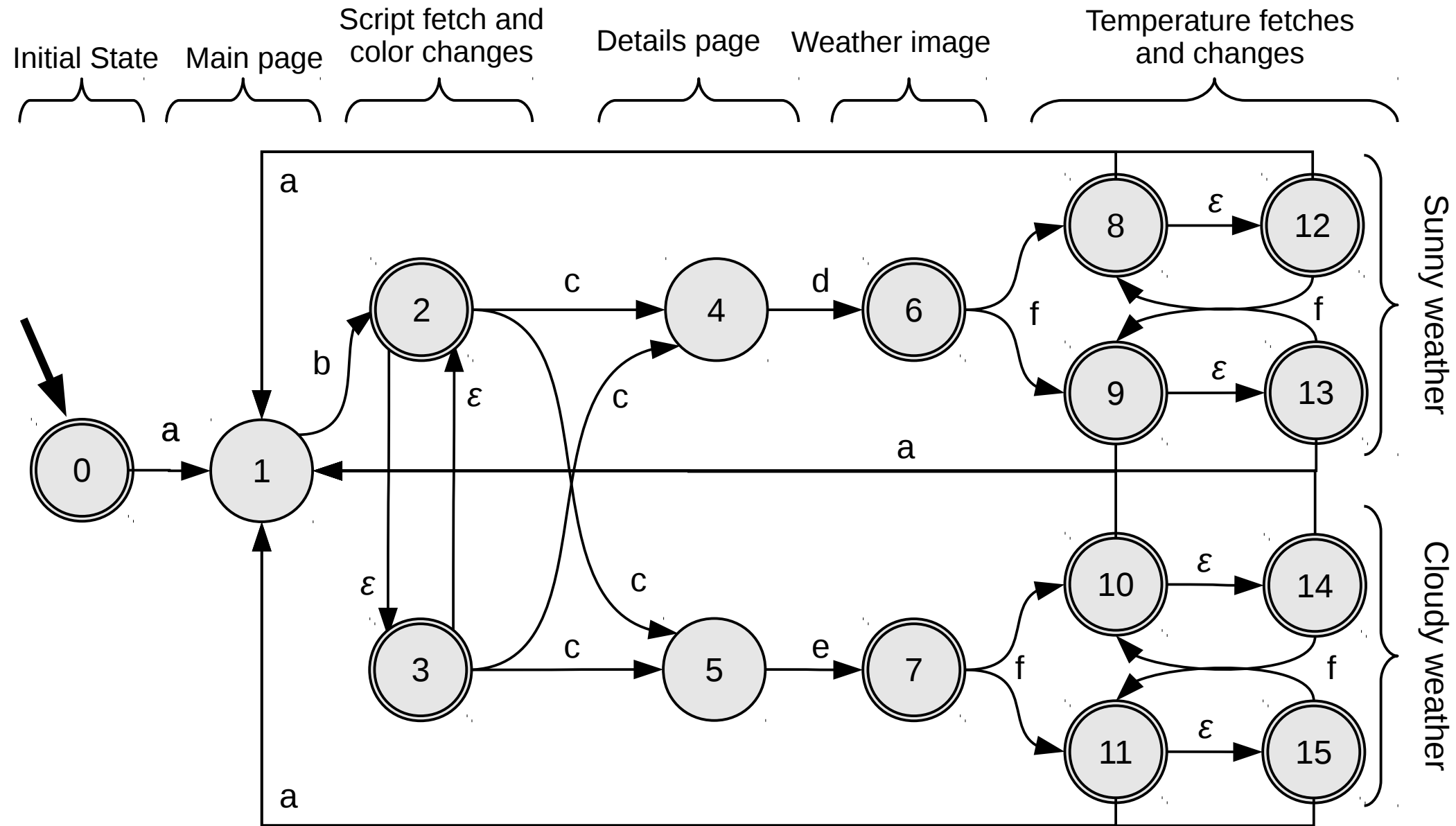
Initial State Main page Script fetch and color changes Details page Weather image



Example: Weather Forecast Web Application



Example: Weather Forecast Web Application



Example: Weather Forecast Web Application

$AS_{\text{main}} = [(\text{metadata} : \dots, \text{data} : \text{"/main contents"}),$
 $(\text{metadata} : \dots, \text{data} : \text{"/script contents"})]$

} **Fully loaded**
/main page

Example: Weather Forecast Web Application

$AS_{\text{main}} = [(\text{metadata} : \text{"..."}, \text{data} : \text{"/main contents"}),$
 $(\text{metadata} : \text{"..."}, \text{data} : \text{"/script contents"})]$

$IS_{\text{toDetails}} = (\text{request} : (\text{method} : \text{"GET"}, \text{resourceId} : \text{"/details"},$
 $\text{representation} : \text{""}), \text{linkType} : \text{"<a>"})$

} **Fully loaded
/main page**

} **/details page
request**

Example: Weather Forecast Web Application

$AS_{\text{main}} = [(\text{metadata} : \dots, \text{data} : \text{/main contents}),$
 $(\text{metadata} : \dots, \text{data} : \text{/script contents})]$

$IS_{\text{toDetails}} = (\text{request} : (\text{method} : \text{GET}, \text{resourceId} : \text{/details},$
 $\text{representation} : \text{""}), \text{linkType} : \text{<a>})$

$\delta(AS_{\text{main}}, IS_{\text{toDetails}}) = \{AS_{\text{detailsCloudy}}, AS_{\text{detailsSunny}}\}$

} **Fully loaded
/main page**

} **/details page
request**

} **/details transition**

Example: Weather Forecast Web Application

$AS_{main} = [(metadata : "...", data : "/main contents"),$
 $(metadata : "...", data : "/script contents")]$

**Fully loaded
/main page**

$IS_{toDetails} = (request : (method : "GET ", resourceId : "/details",$
 $representation : ""), linkType : "<a>")$

**/details page
request**

$\delta(AS_{main}, IS_{toDetails}) = \{AS_{detailsCloudy}, AS_{detailsSunny}\}$

/details transition

$AS_{detailsCloudy} = [(metadata : [mediaType : "text/html"],$
 $data : "/details content with link to /cloudy")]$

**Partially loaded
/details page**

$AS_{detailsSunny} = [(metadata : [mediaType : "text/html"],$
 $data : "/details content with link to /sunny")]$

Future Work

- Reduce state explosion problem
 - Aggregating similar states into a single state
 - States are similar if they have the same representations with the same links, but data may be different

Future Work

- Reduce state explosion problem
 - Aggregating similar states into a single state
 - States are similar if they have the same representations with the same links, but data may be different
- Apply the formalism to more systems
 - Web 1.0 documents vs Web 2.0 applications
 - Web APIs vs Web applications

Future Work

- Reduce state explosion problem
 - Aggregating similar states into a single state
 - States are similar if they have the same representations with the same links, but data may be different
- Apply the formalism to more systems
 - Web 1.0 documents vs Web 2.0 applications
 - Web APIs vs Web applications
- Unaddressed principles of RESTful systems
 - Layered and cacheable constraints

Future Work

- Reduce state explosion problem
 - Aggregating similar states into a single state
 - States are similar if they have the same representations with the same links, but data may be different
- Apply the formalism to more systems
 - Web 1.0 documents vs Web 2.0 applications
 - Web APIs vs Web applications
- Unaddressed principles of RESTful systems
 - Layered and cacheable constraints
- Software framework for development of RESTful systems

Conclusion

- Understanding REST *is important*
 - Formal models, systematization, terminology
- eNFA formalism
 - Captures “~98%” of REST's principles
 - Simple, generic, operational, system-wide
- Exciting directions for future research!
 - Anonymous reviewer: “*A model by itself has little value unless it is used for some purpose.*”



2011 ... Paphos, Cyprus

2011 ... Paphos, Cyprus

2011 ... Paphos, Cyprus

2011 ... Paphos, Cyprus

Thank you!

Contact:

izuzak@gmail.com

<http://twitter.com/izuzak>



2011 ... Paphos, Cyprus

2011 ... Paphos, Cyprus

2011 ... Paphos, Cyprus

2011 ... Paphos, Cyprus