Flexible techniques for analyzing and manipulating web service protocols

Farouk Toumani

27 June 2006

ISIMA, Université Blaise Pascal – Clermont-Ferrand II

LIMOS, Lab. d'Informatique, de Modélisation et d'Optimisation des Systèmes

Material based on work in **ServiceMosaic** project (http://servicemosaic.isima.fr)

Collaboration with *colleagues and students*

- Prof. Boualem Benatallah (UNSW, Australia)
- Dr. Fabio Casati (HP Labs, USA)
- Mr. Hamid Motahari (UNSW, Australia)
- Mr. Julien Ponge (LIMOS, France)

Agenda

- Web services vision and technologies
- Representing web service protocols
- Analysis and management of web service protocols
- Summary and outlook

Motivations

- Integration of autonomous and heterogeneous systems
- Automation of inter-organizational business processes
- Web services : evolution of current technologies
  - Distributed information systems
  - Middleware (RPC, MOM, CORBA,),
  - Entreprise Application Integration

Beyond current technologies

- New integration context
  - Open environment : autonomous systems
  - Large and dynamic integration space
  - Semantic heterogeneity (both data and business processes): one-to-one mappings between partner systems do not scale
  - Inter-organizational interactions (trust, security, transactions, etc)
- Limitations of current technologies
  - Centralized middleware
  - Rigid infrastructures, costly development and maintenance of integrated systems
  - Close environment/tightly coupled systems (semantics known from the context)

Web services

“a software application identified by a URI, whose interfaces and bindings are capable of being defined, described and discovered as XML artifacts. A web service supports direct interactions with other software agents using XML-based messages exchanged via Internet-based protocols” [W3C]

Main characteristics

- Generic interface for service oriented architectures
- Intensive use of standards (SOAP, WSDL, etc)
- Loosely coupled integration

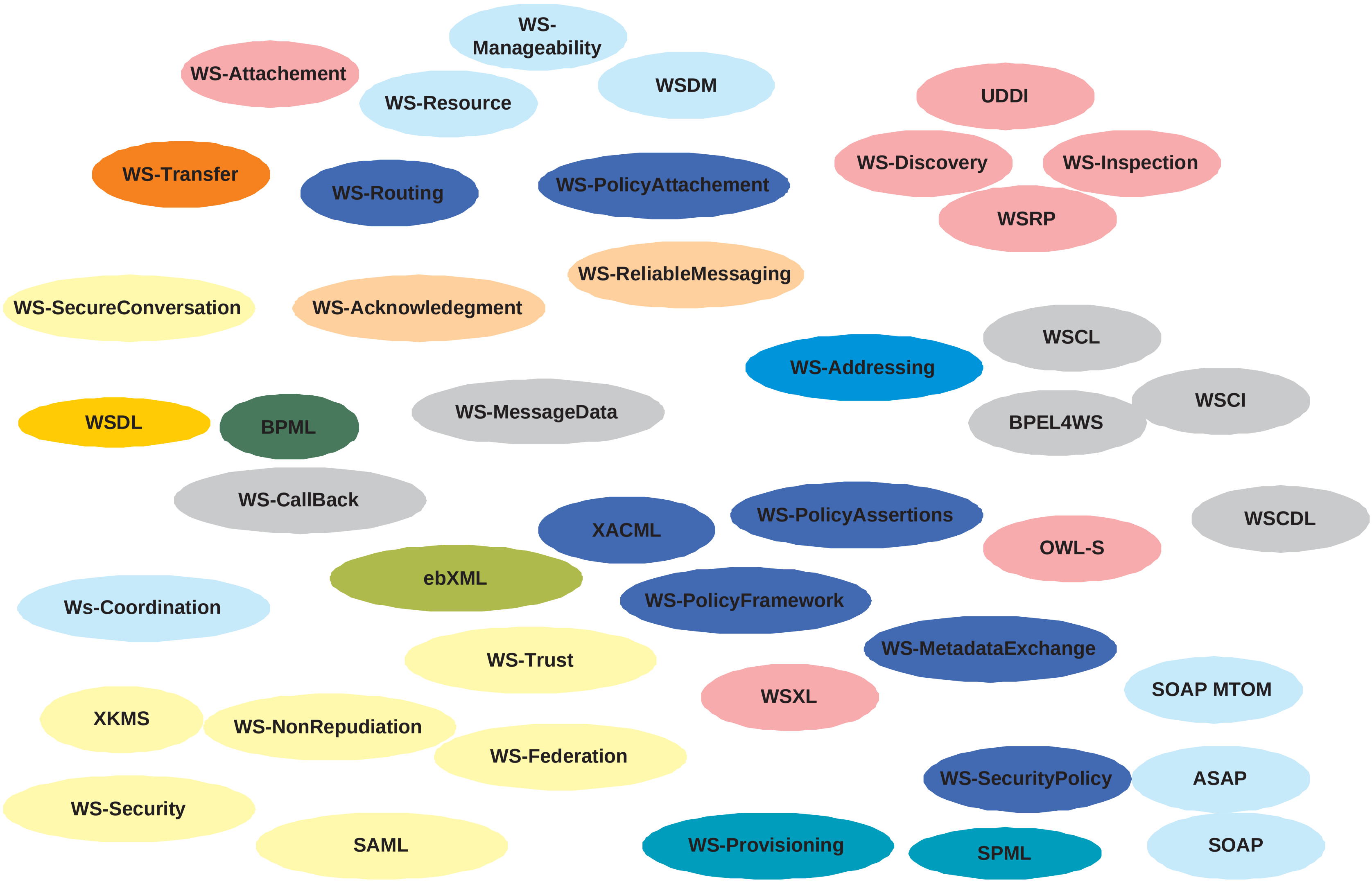
⇒ **Ultimate goal:** rapid low-cost development and easy composition of distributed applications

Web service technologies

Specifications and languages providing **core functionality** of web services

- Service description
- Service discovery
- Service interactions
- Service composition

Current Standards puzzle

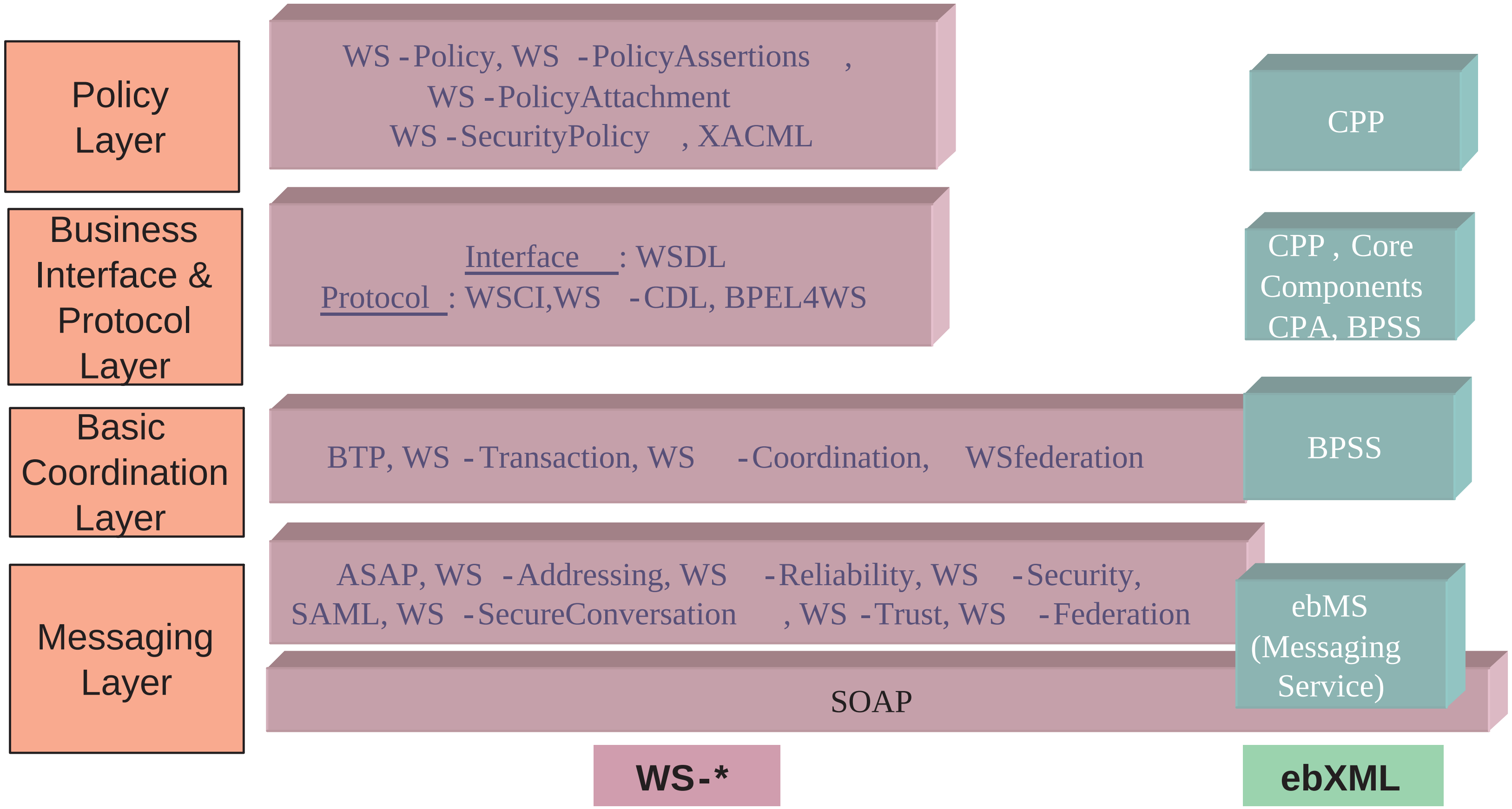


WS-Manageability
WS-Attachement
WS-Resource
WSDM
UDDI
WS-Transfer
WS-Routing
WS-PolicyAttachement
WS-Discovery
WS-Inspection
WSRP
WS-ReliableMessaging
WS-SecureConversation
WS-Acknowledegment
WSCL
WS-Addressing
WSCI
WSDL
BPML
WS-MessageData
BPEL4WS
WS-CallBack
XACML
WS-PolicyAssertions
WSCDL
OWL-S
ebXML
Ws-Coordination
WS-PolicyFramework
WS-MetadataExchange
WS-Trust
WSXL
SOAP MTOM
XKMS
WS-NonRepudiation
WS-Federation
WS-SecurityPolicy
ASAP
WS-Security
SAML
WS-Provisioning
SPML
SOAP

Interoperability layers

Policy Layer	**Policy specification (e.g., privacy policies) and non-functional properties (e.g., cost, response time, ..)**
Business Interface & Protocol Layer	**Functional properties of services (interfaces and business protocols)**
Basic Coordination Layer	**Requirements and properties related to a set of message exchanges among two or more partners**
Messaging Layer	**Standard information transportation protocol**

Interoperability layers



Policy Layer

WS -Policy, WS -PolicyAssertions , WS -PolicyAttachment WS -SecurityPolicy , XACML

CPP

Business Interface & Protocol Layer

Interface : WSDL
Protocol : WSCI,WS -CDL, BPEL4WS

CPP , Core Components CPA, BPSS

Basic Coordination Layer

BTP, WS -Transaction, WS -Coordination, WSfederation

BPSS

Messaging Layer

ASAP, WS -Addressing, WS -Reliability, WS -Security, SAML, WS -SecureConversation , WS -Trust, WS -Federation

ebMS (Messaging Service)

SOAP

WS-*

ebXML

Some Observations

- Services are loosely-coupled and need to be fully specified

  Making implicit information (as in closed environments) explicit (essential in autonomous environments)

  – Interface,

  – Business protocols,

  – Functional and non-functional properties (e.g., QoS, . . .),

  – Meaning of the parameters, operations effects, negociation parameters, . . .

→ Trade-off: expressive power vs. readability/usability

Agenda

- Web services vision and technologies
- **Representing web service protocols**
- Analysis and management of web service protocols
- Summary and outlook

ServiceMosaic project

Joint project with SOC group (UNSW, Sydney) and HP laboratories (Palo Alto, USA)

- Definition of a service description framework

  A protocol model endowed with richer abstractions and a formal semantics

- Design of an algebra for high level analysis and management of service protocols
- Protocol discovery
- Model-driven approach to support service adaptation
- Model-driven change impacts analysis
- Development of a fully-fledged CASE tool for Web service development and lifecycle management

Protocol modeling

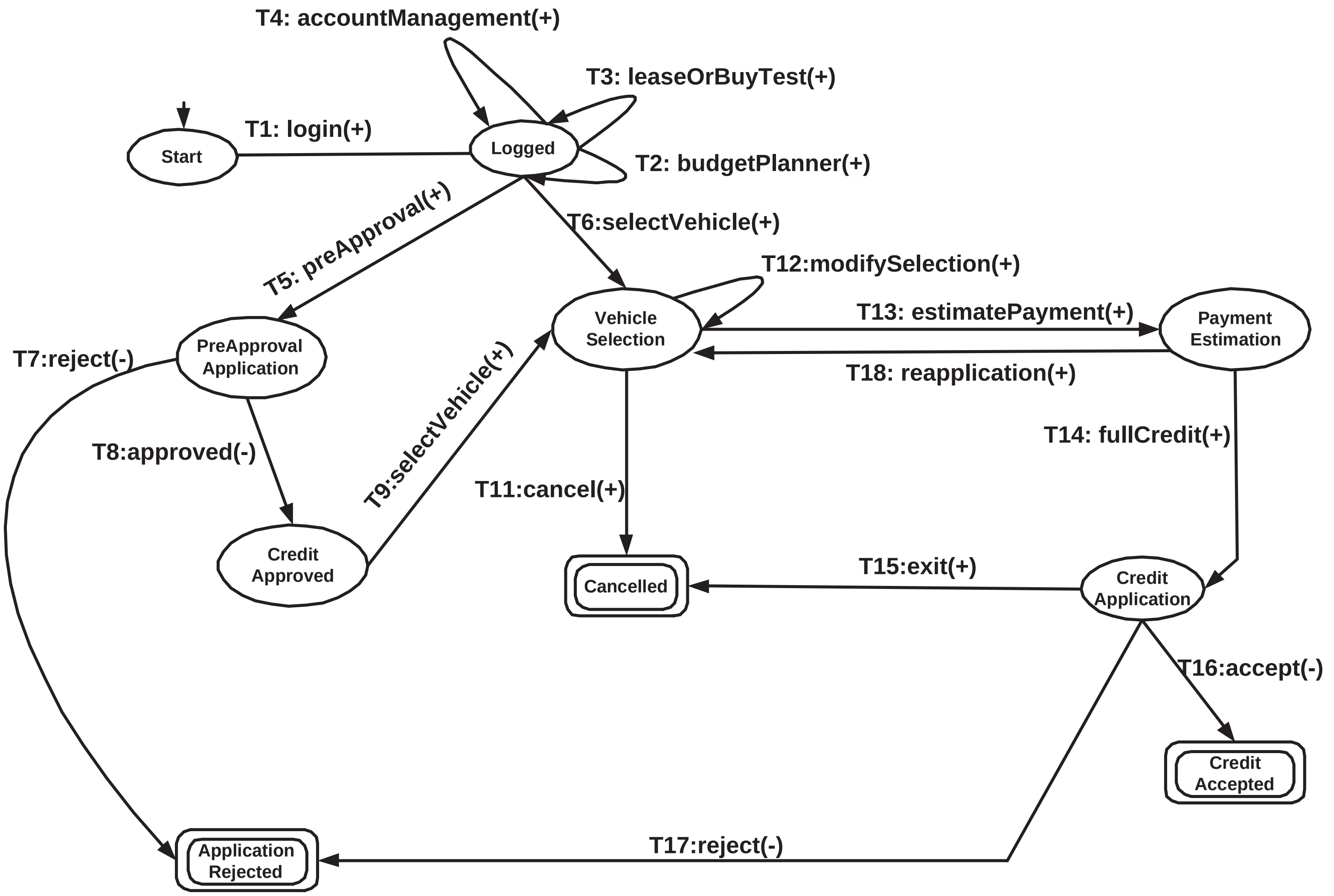
Describe external behavior of services

An extended protocol model

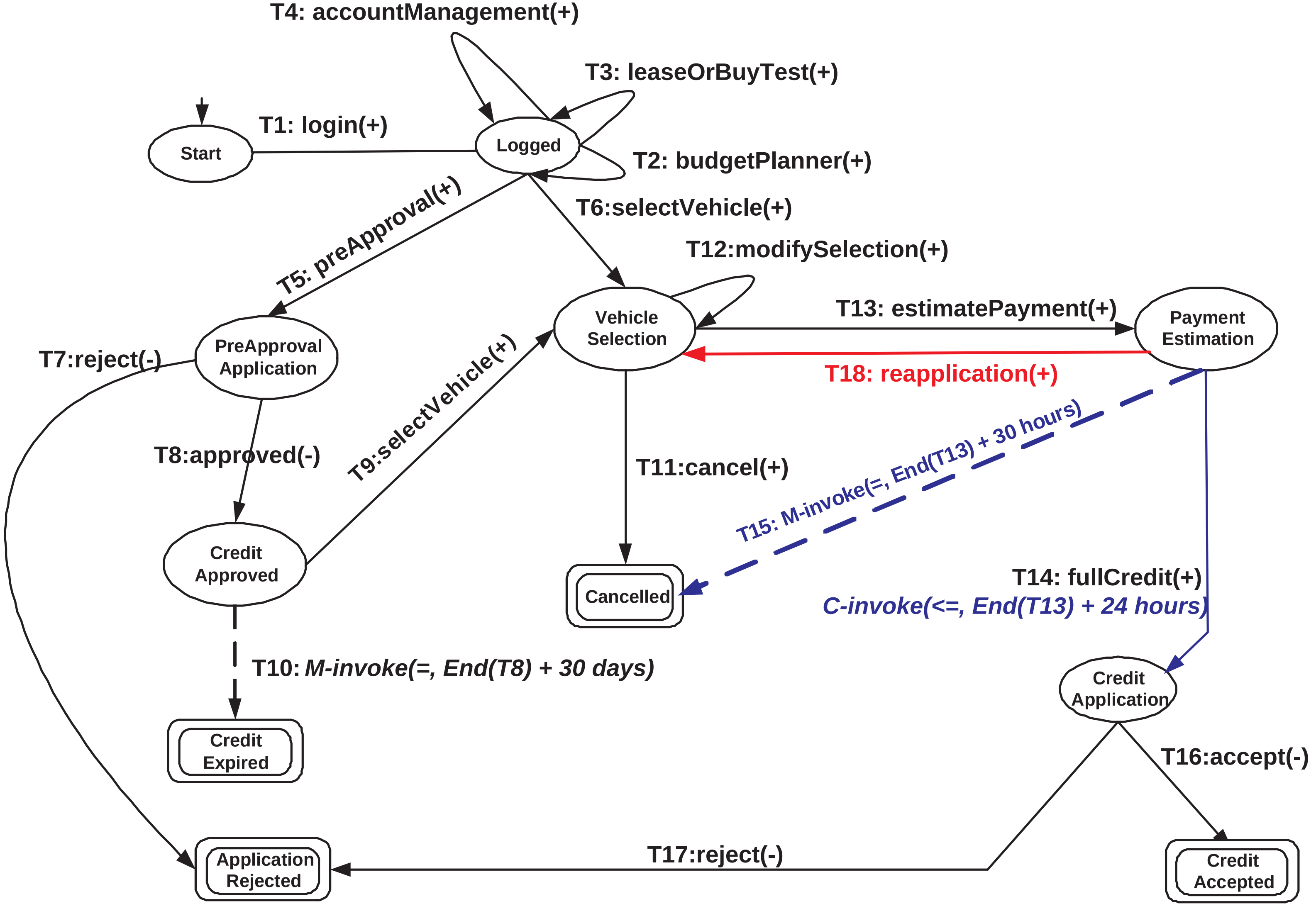
- Message Choreography
- Time-sensitive Conversations
- Transactional Implications and Effects

Rightsizing is a key issue

Example: a credit online financing services



Example: a credit online financing services (cont.)



Protocol analysis and management

- Two dimensions of the analysis
  - Compatibility: checking whether two services can interact correctly based on their protocol definitions
  - Replaceability: verifying whether two protocols can support the same set of conversations
- Characterization of different levels of protocol compatibility and replaceability
- Need for a protocol algebra
  - primitives to analyze and manage protocols
  - key to achieve the benefits
  - tools that implement these operators

⇒ Provide an automated support for verification of compatibility and replaceability between pairs of protocols

Applications and benefits

- Service discovery and composition (e.g., reduce the number of *false positives* during service discovery)
- Change support and evolution
- Support for static and dynamic binding
- Compliance verification (e.g., with B2B standards)

Formalization

!! Small is beautiful

A basic protocol: message choreography + message polarity

- Protocol model based on a finite state machine
- Protocol semantics : branching time view

  Protocol $\mathcal{P}$ = a schema (i.e., intentional description of service behavior)

  Semantics of a protocol: set of *complete execution trees*

  Semantics of interactions : set of *complete interaction trees*

  ⇒ Useful to deal with service composition
- A slightly adapted simulation relation to compare protocols

Compatibility classes

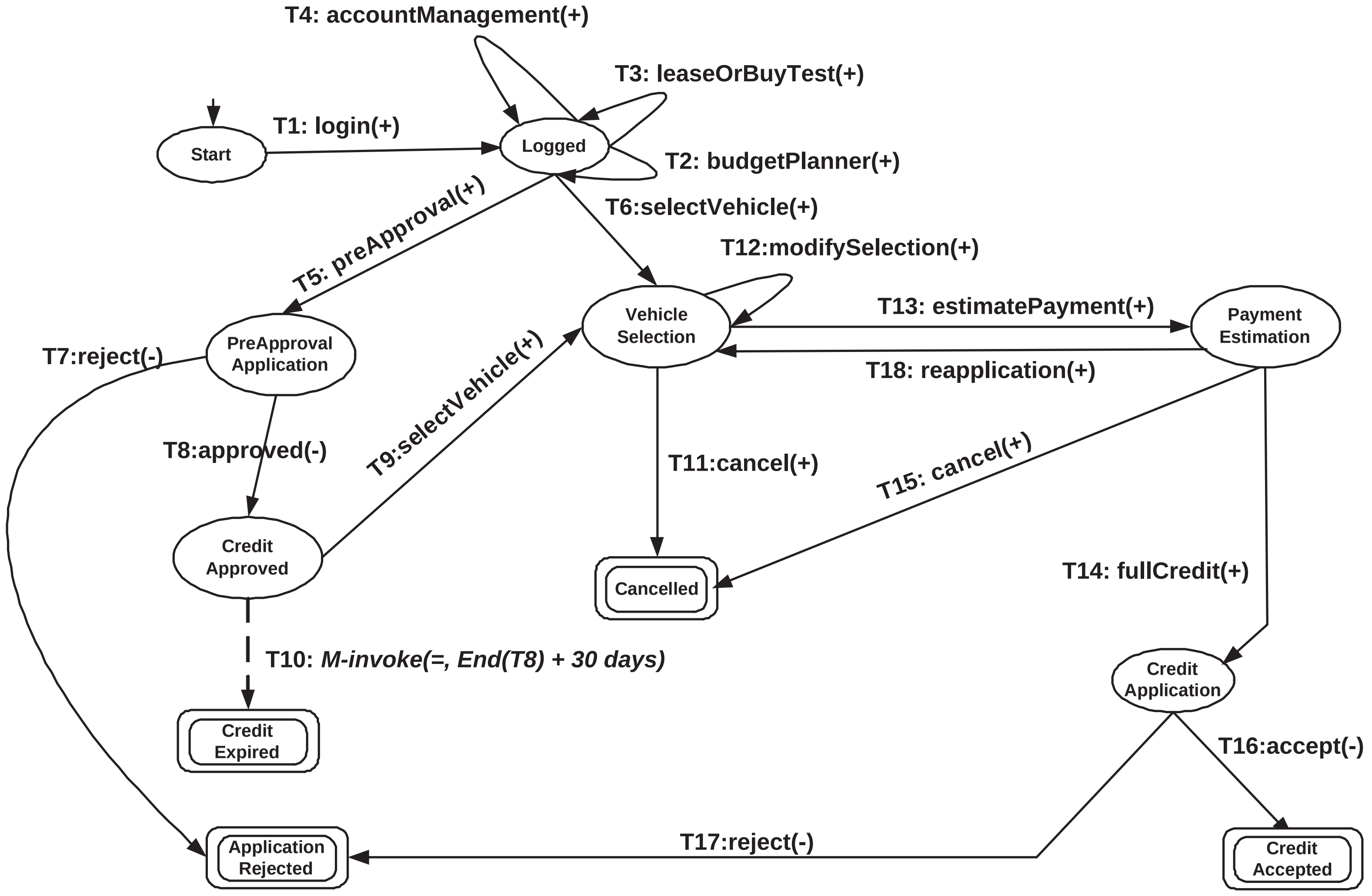
- Partial compatibility

  $\mathcal{P}_1$ is partially compatible with $\mathcal{P}_2$ if there are some *executions* of $\mathcal{P}_1$ that can interoperate with $\mathcal{P}_2$, i.e., if there is at least one possible conversation that can take place

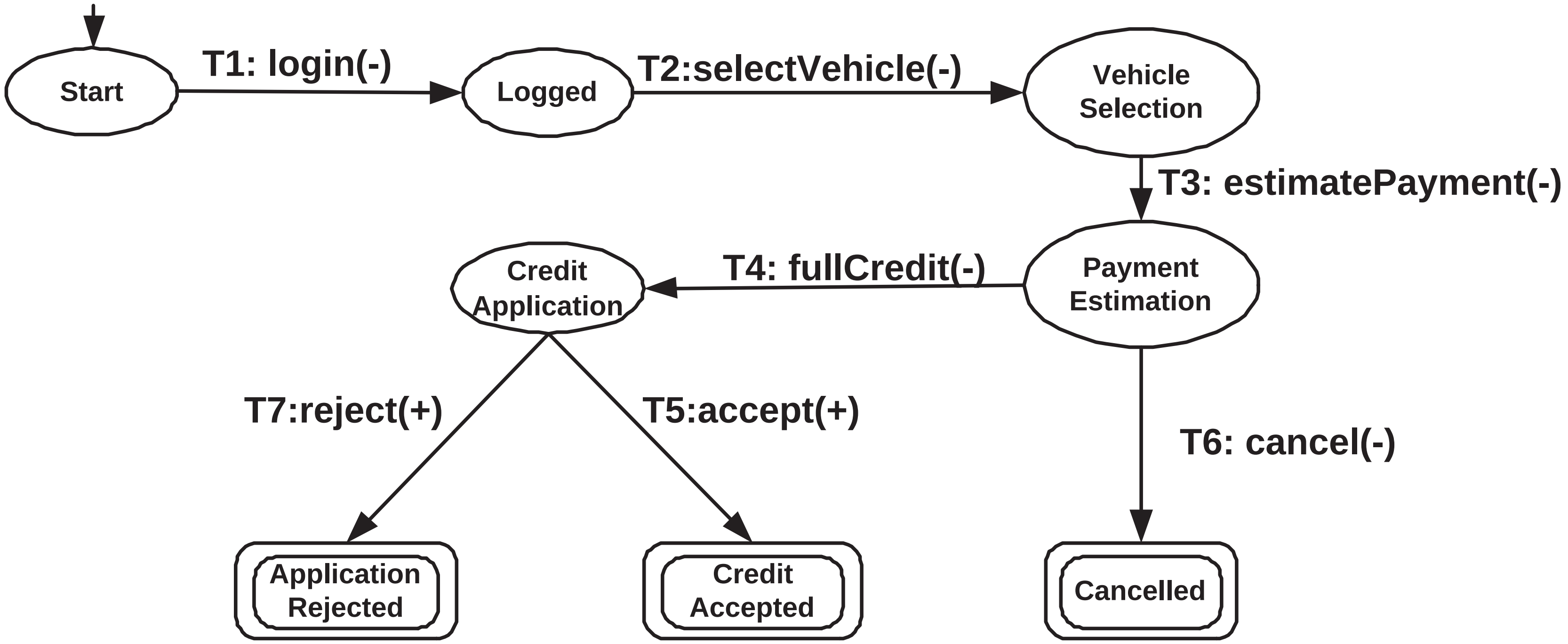
- Full compatibility

  $\mathcal{P}_1$ is fully compatible with $\mathcal{P}_2$ if all executions of $\mathcal{P}_1$ can interoperate with $\mathcal{P}_2$, i.e., any conversation that can be generated by $\mathcal{P}_1$ is understood by $\mathcal{P}_2$

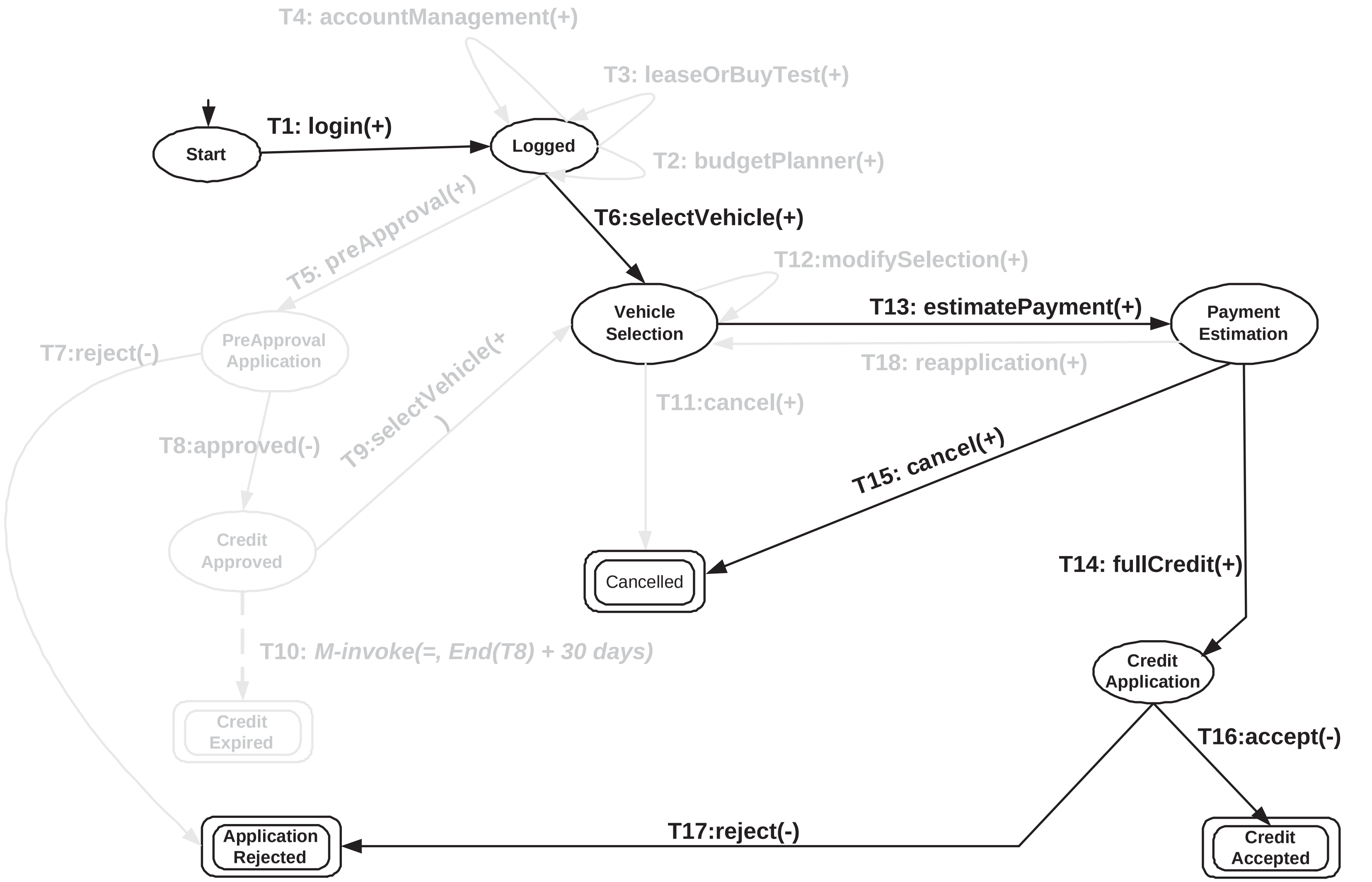
Example: a basic online financing services



Example: a simple client protocol $\mathcal{P}_C$



Example: compatibility



Replaceability classes

- Protocol equivalence

  Identifies when two protocols can be interchangeably used in any context and the change is transparent to clients

- Protocol subsumption

  Identifies when a protocol $\mathcal{P}_1$ can be transparently used instead of $\mathcal{P}_2$ (the opposite is not necessarily true)

- Protocol equivalence and subsumption with respect to a client protocol

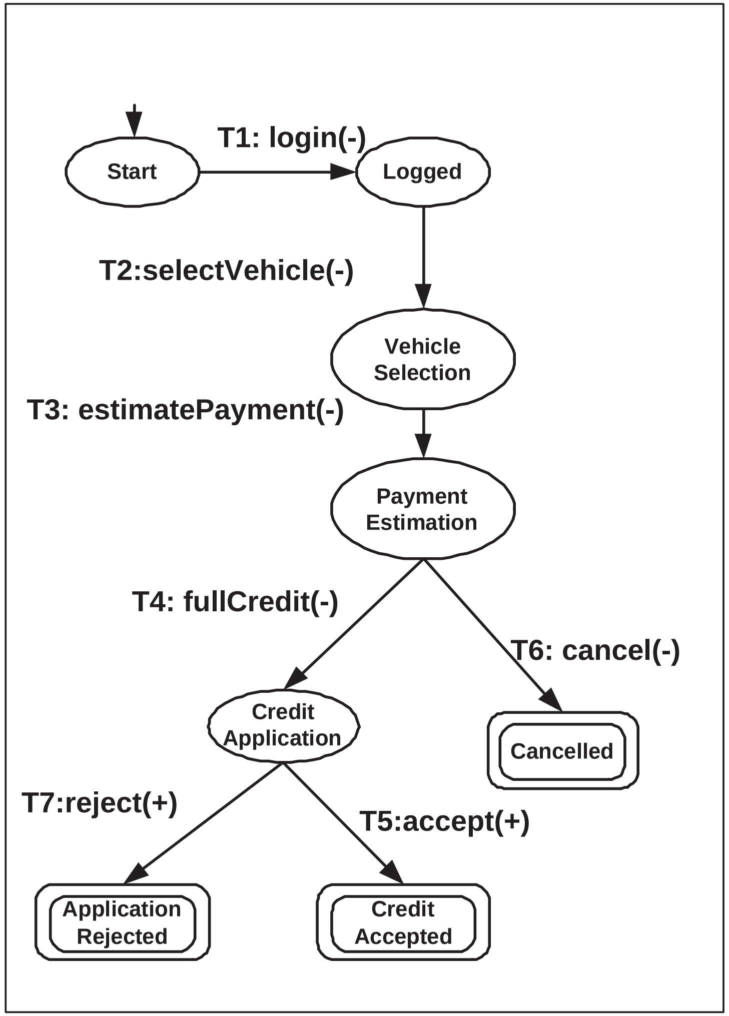
  Identifies replaceability relations with respect to a certain client

- Protocol equivalence and subsumption with respect to an interaction role

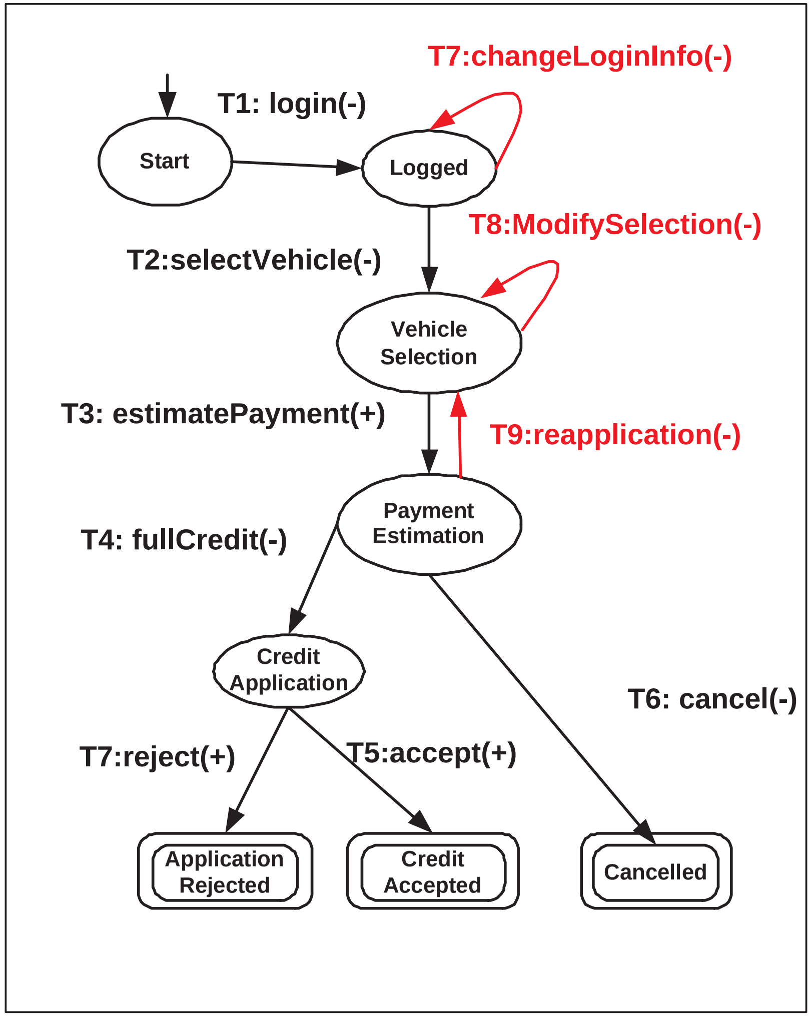
  $\mathcal{P}_2$ can replace $\mathcal{P}_1$ with respect to a role $\mathcal{P}_R$ if $\mathcal{P}_2$ behaves as $\mathcal{P}_1$ when $\mathcal{P}_1$ behaves as $\mathcal{P}_R$

Example: subsumption

P1

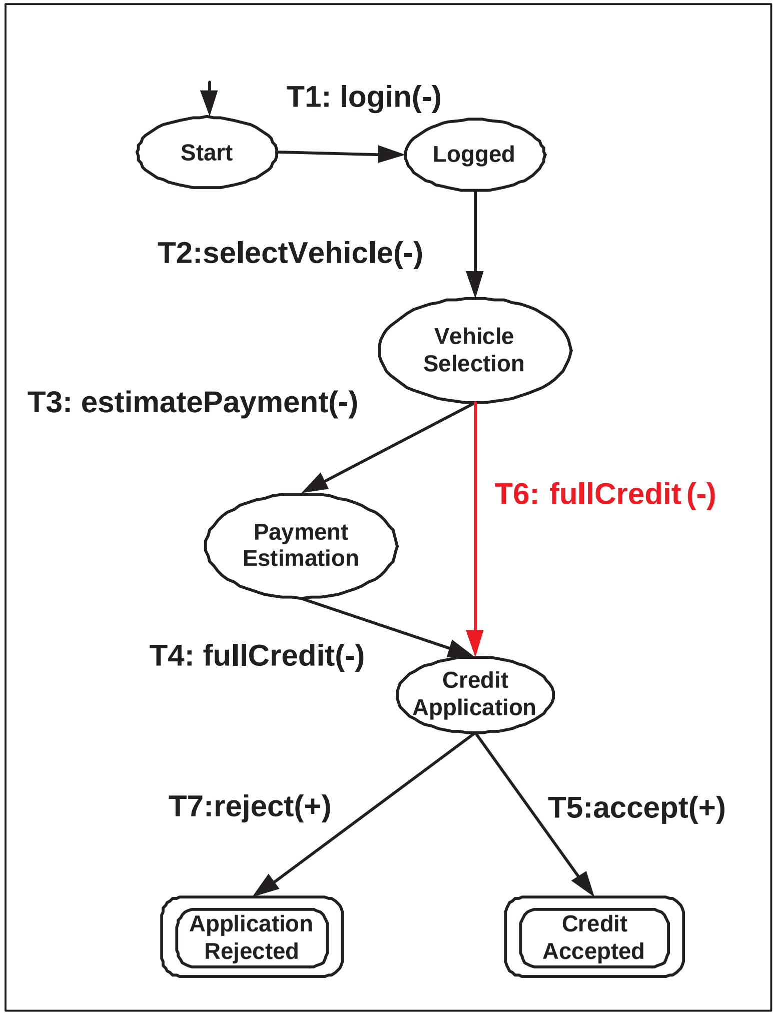


P2

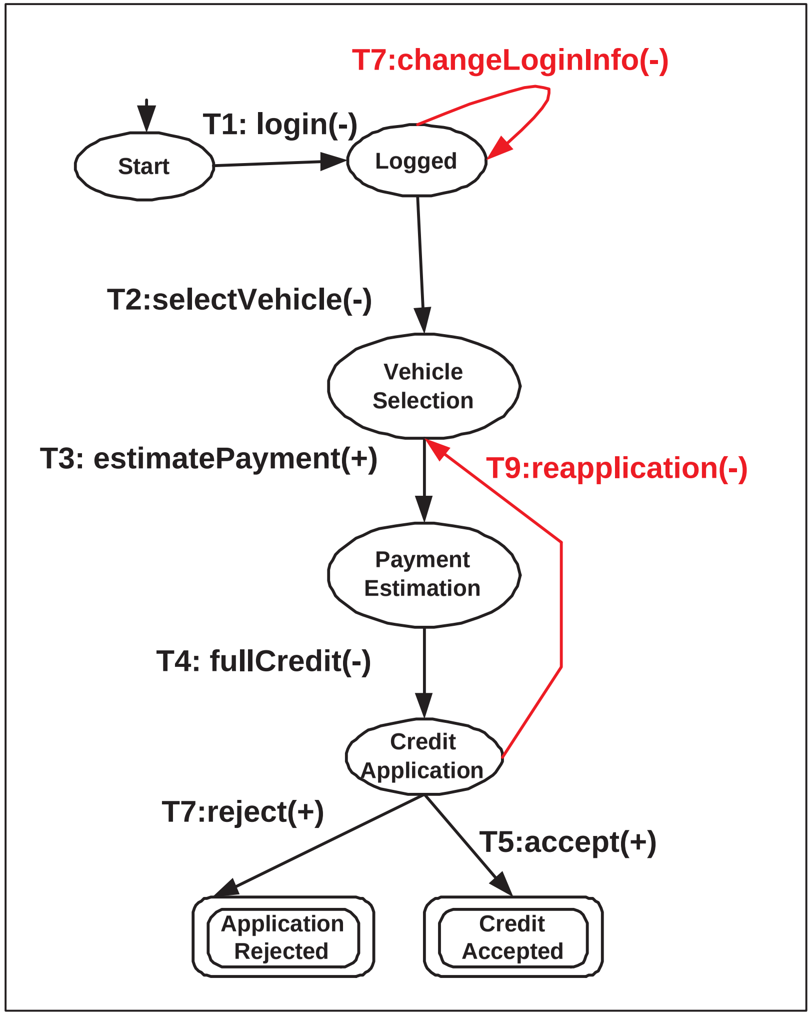


Example: equivalence w.r.t. a client protocol

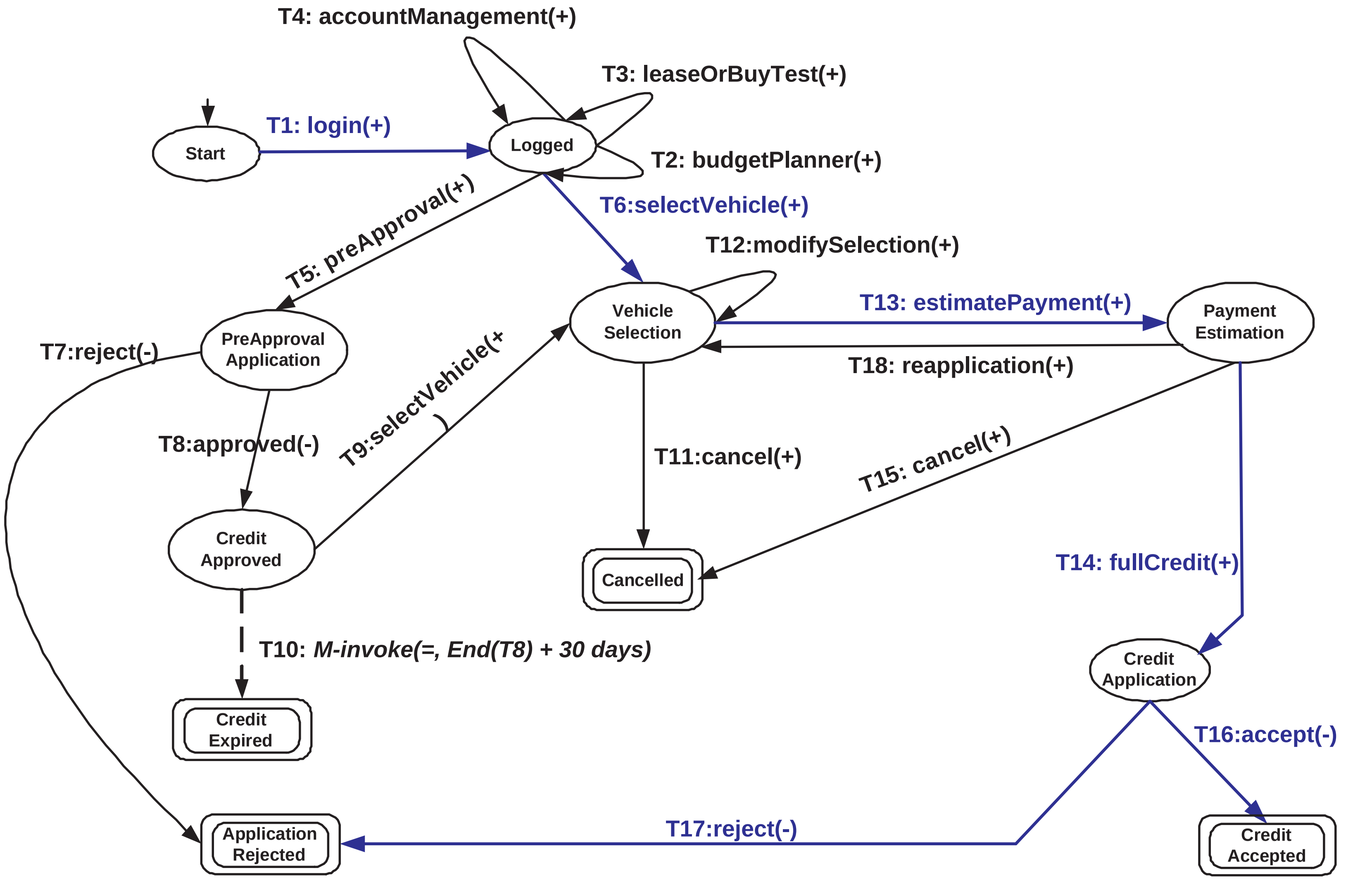
P3



P4



Example: equivalence w.r.t. a client protocol (cont.)



Toward a protocol algebra

A set of operators to manipulate and analyze protocols

Example

- Compatible composition

  Returns a protocol that describes all the possible conversations between the considered protocols

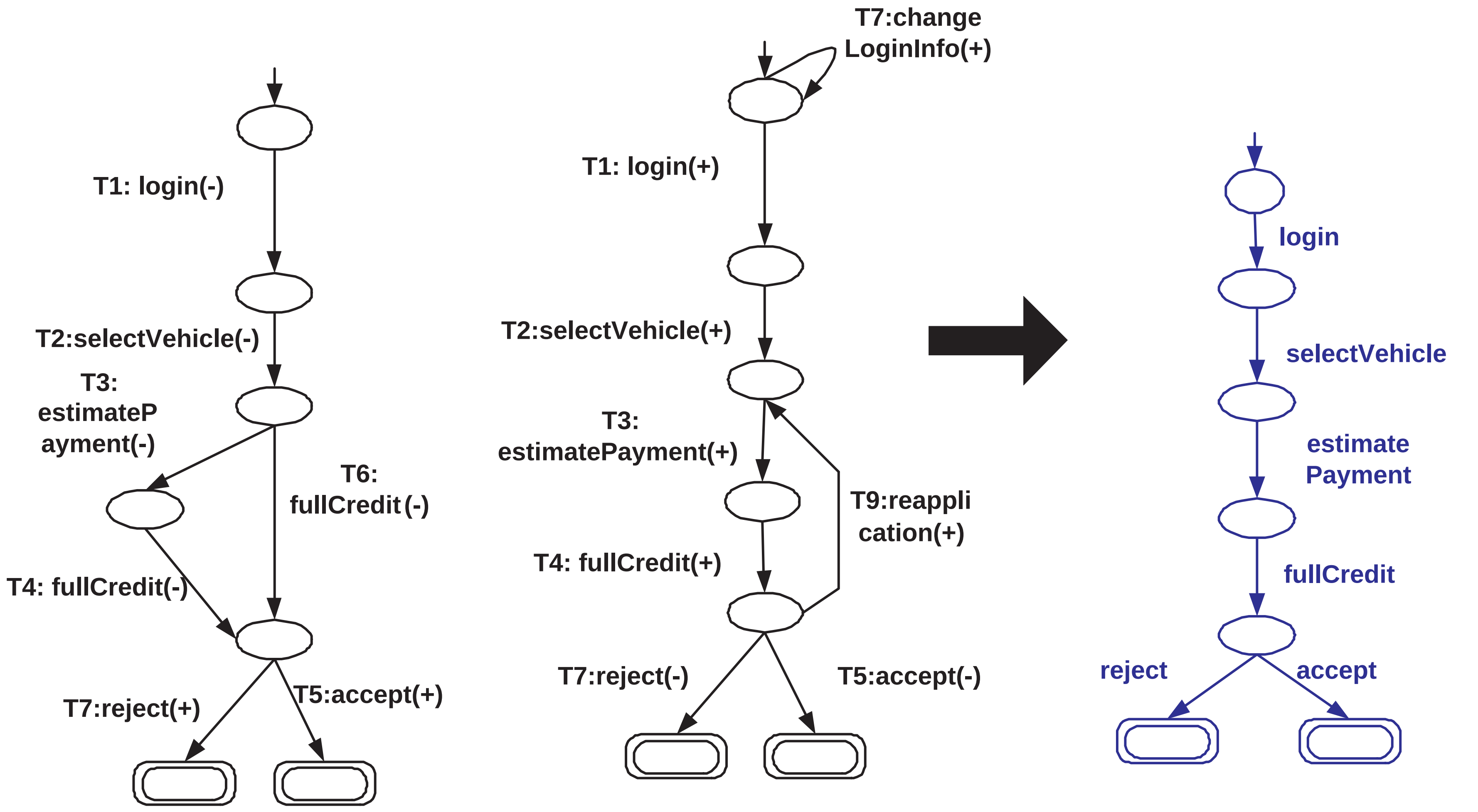
- Intersection

  Returns a protocol that describes the set of conversations that are common between the input protocols

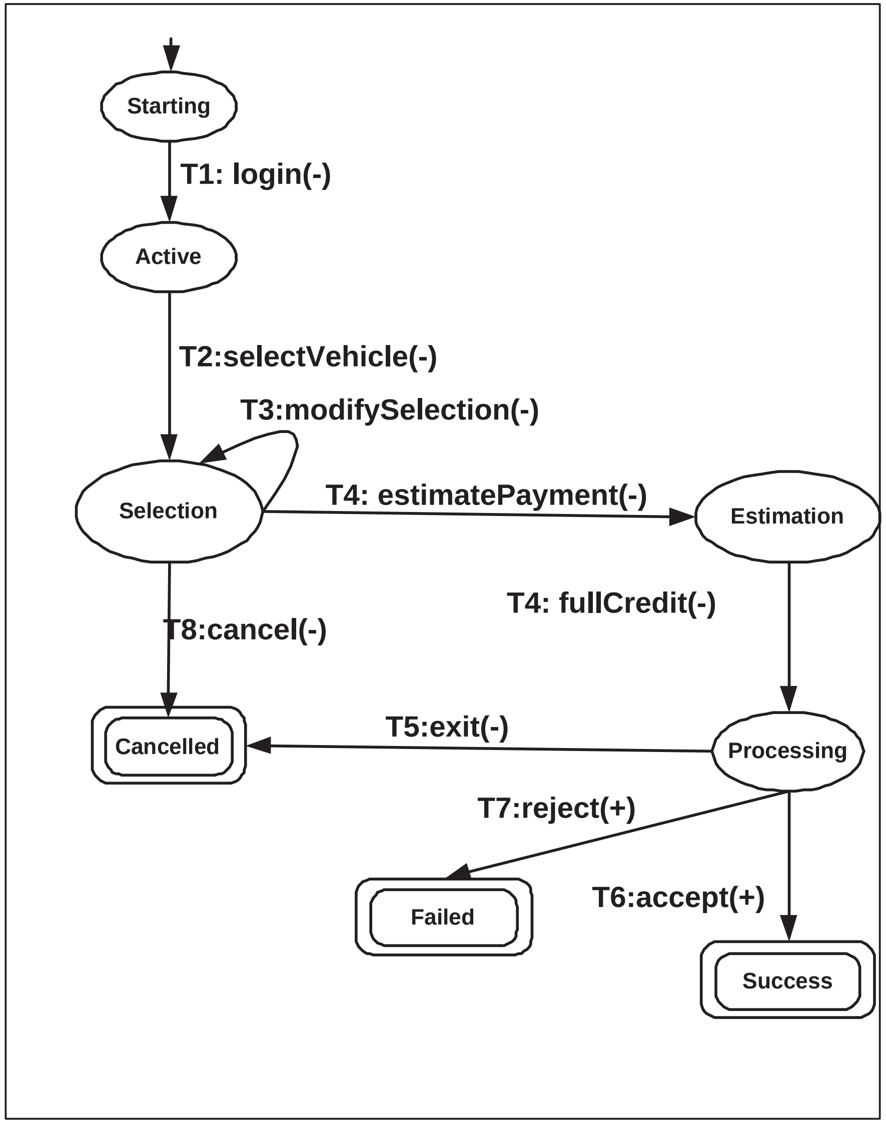
- Difference

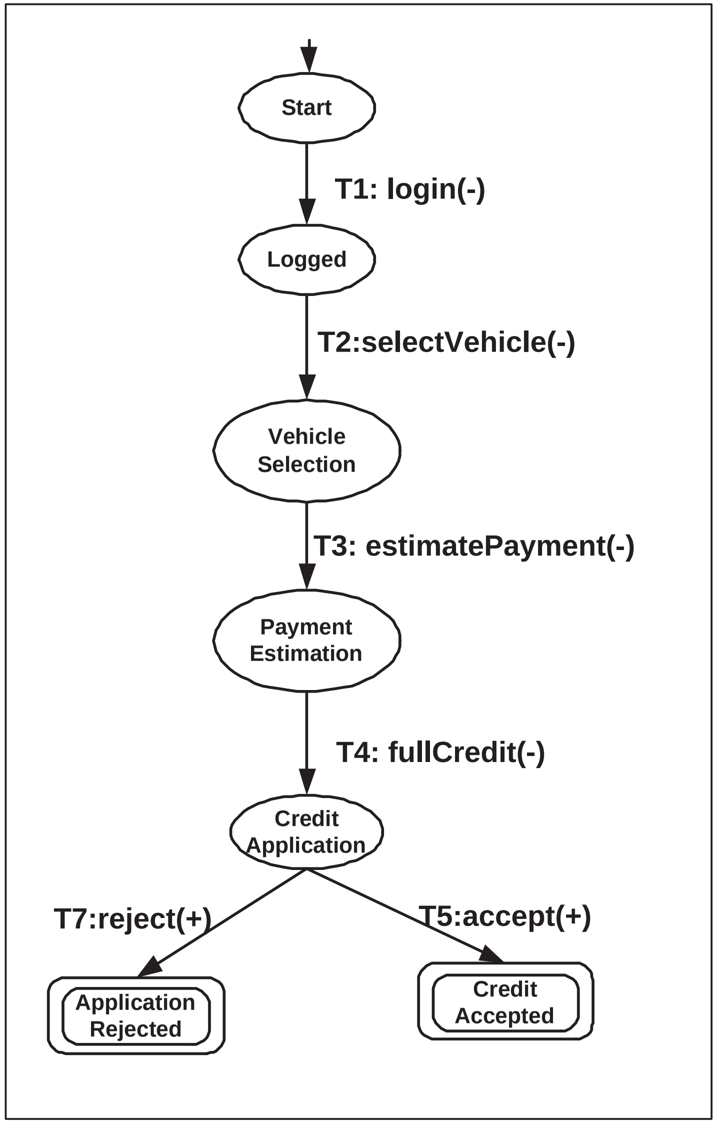
  Returns a protocol that describes the set of conversations of the first input protocol that cannot be supported by the second input protocol

Example: compatible composition

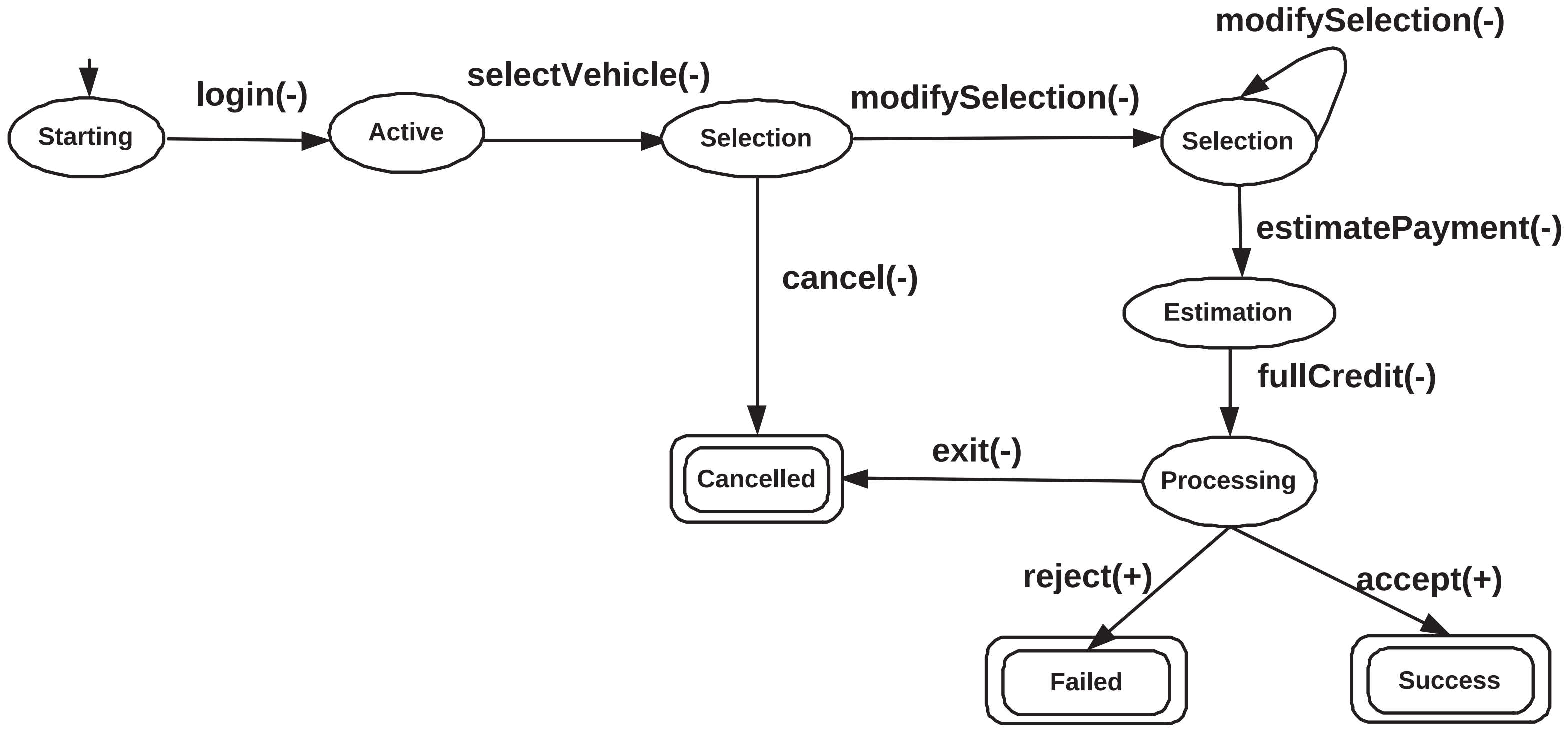


Example: difference operation





Example: difference operation (cont.)



Characterization of the classes

- **Partial compatibility**

  $P\text{-}compat(\mathcal{P}_1, \mathcal{P}_2)$ iff $\mathcal{P}_1 ||^{\mathtt{C}} \mathcal{P}_2$ is not an empty protocol

- **Full compatibility**

  $F\text{-}compat(\mathcal{P}_1, \mathcal{P}_2)$ iff $[\mathcal{P}_1 ||^{\mathtt{C}} \mathcal{P}_2]_{\mathcal{P}_1} \cong \mathcal{P}_1$.

- **Subsumption**

  $Subs(\mathcal{P}_2, \mathcal{P}_1)$ iff $\mathcal{P}_2 \lesssim \mathcal{P}_1$

- **Equivalence**

  $Equiv(\mathcal{P}_1, \mathcal{P}_2)$ iff $\mathcal{P}_1 \cong \mathcal{P}_2$

- **Replaceability with respect to a client protocol**

  $Repl_{[\mathcal{P}_c]}(\mathcal{P}_1, \mathcal{P}_2)$ iff $\mathcal{P}_c ||^{\mathtt{C}} (\mathcal{P}_2 ||^{\mathtt{D}} \mathcal{P}_1)$ is an empty protocol

- **Replaceability with respect to an interaction role**

  $Repl\_Role_{[\mathcal{P}_R]}(\mathcal{P}_1, \mathcal{P}_2)$ iff $(P_R ||^{\mathtt{I}} \mathcal{P}_2) \lesssim \mathcal{P}_1$.

Extension to timed protocols

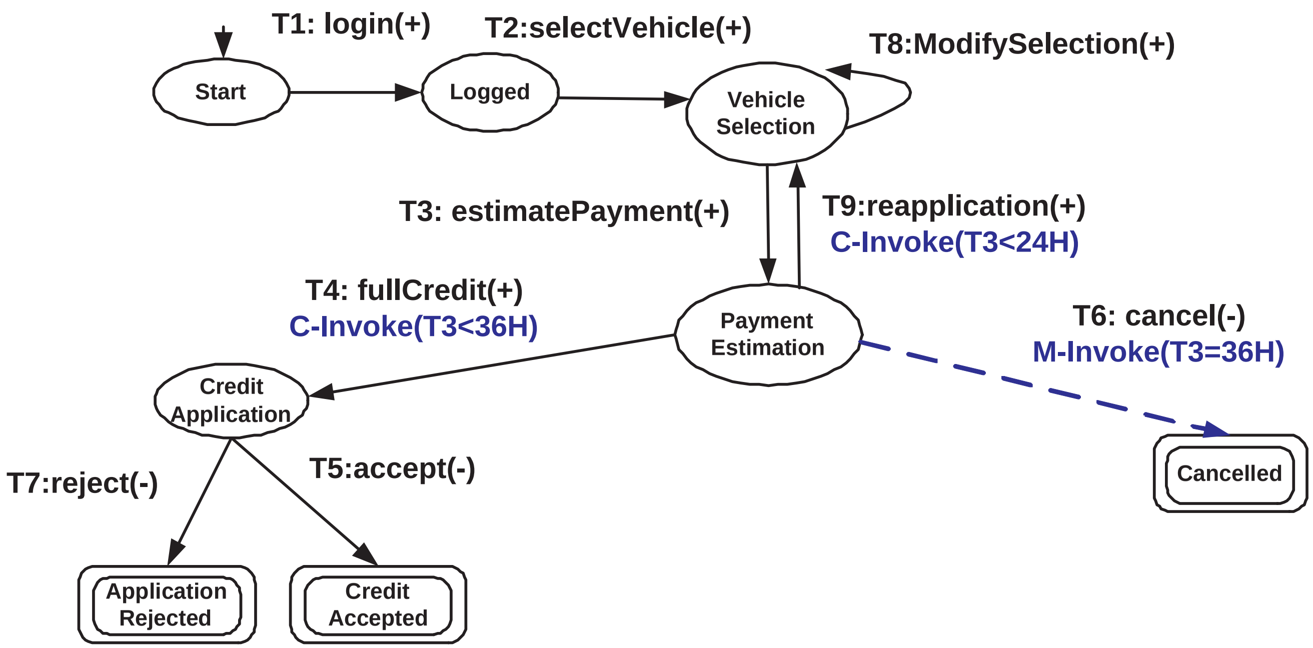
Timed protocol = basic protocol + temporal abstractions

- Two main temporal abstractions
  - C-invoke: temporal window
  - M-Invoke: expiration
- Formal semantics

  Protocol semantics : set of timed execution paths (conversation)

  Interaction semantics : set of timed interaction paths

Example of a timed protocol

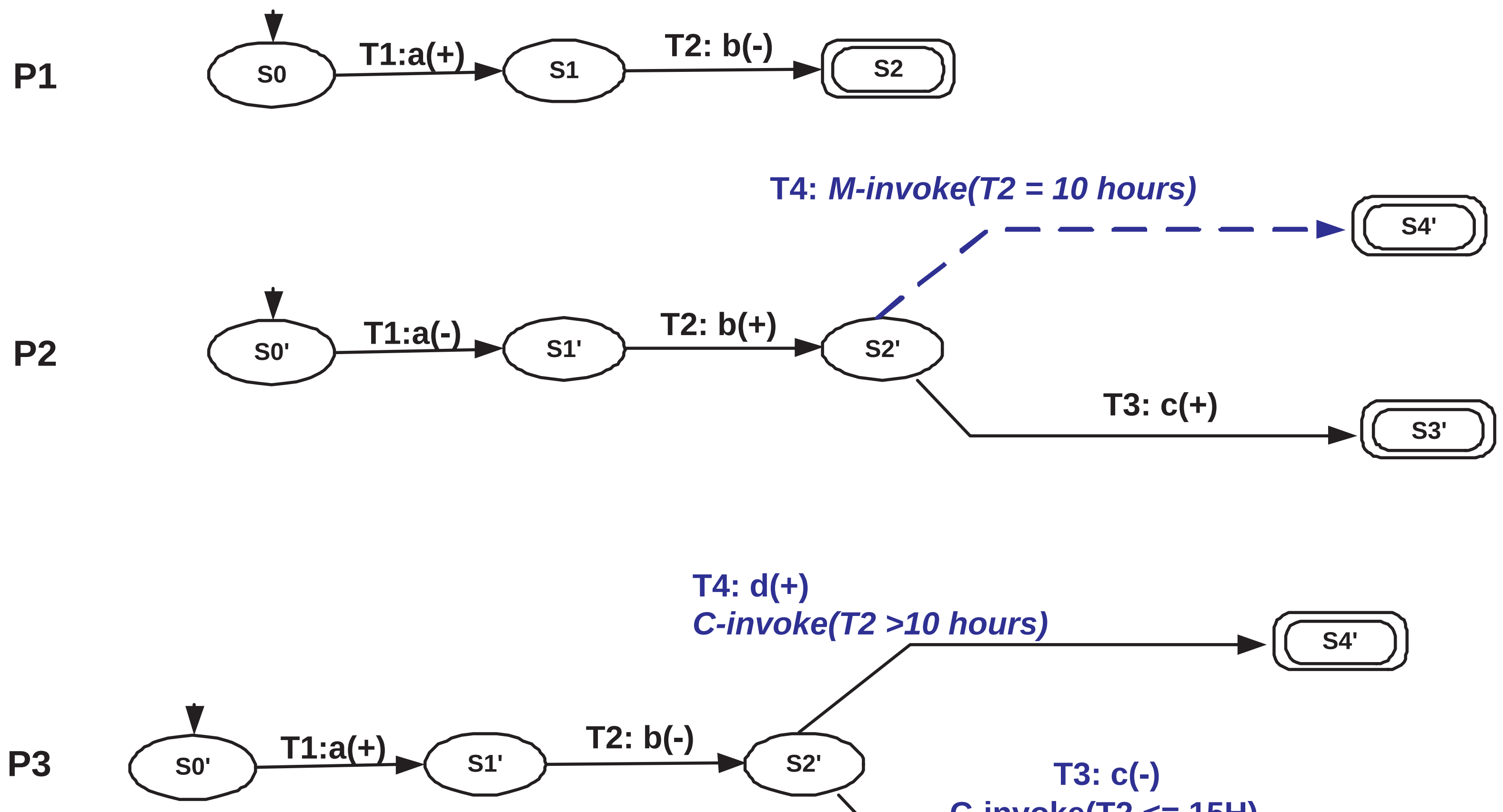


Timed conversation: *(login(+),0);(selectVehicle(+),1); (estimatePayment(+), 2);(cancel(-),40)*

Analysis and management of timed protocols

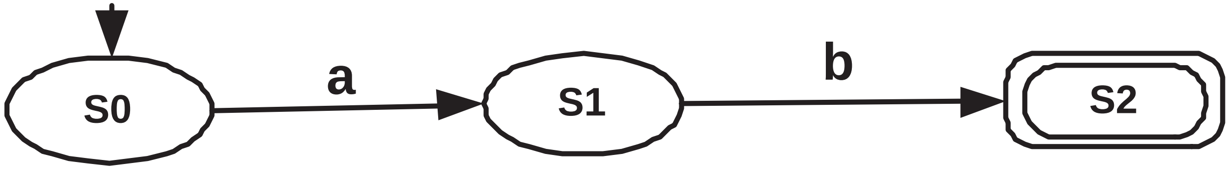
- Definition of new time-sensitive replaceability and compatibility classes
- Definition of operators for analyzing and managing timed protocols

Example: time-sensitive compatibility

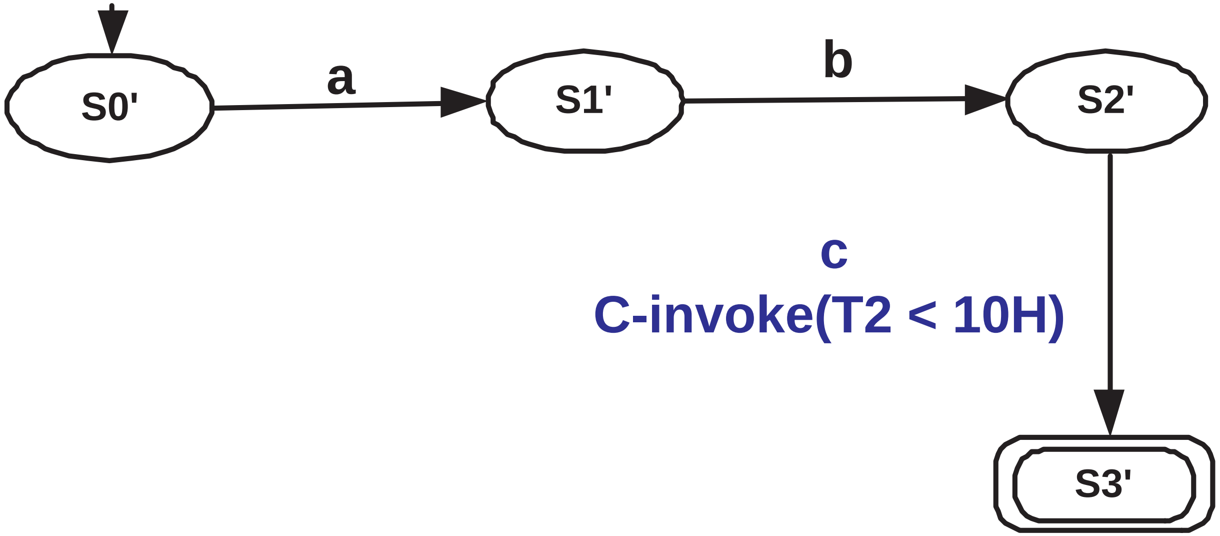


Example: time compatible composition

$P1 \;||^{TC}\; P2$



$P2 \;||^{TC}\; P3$



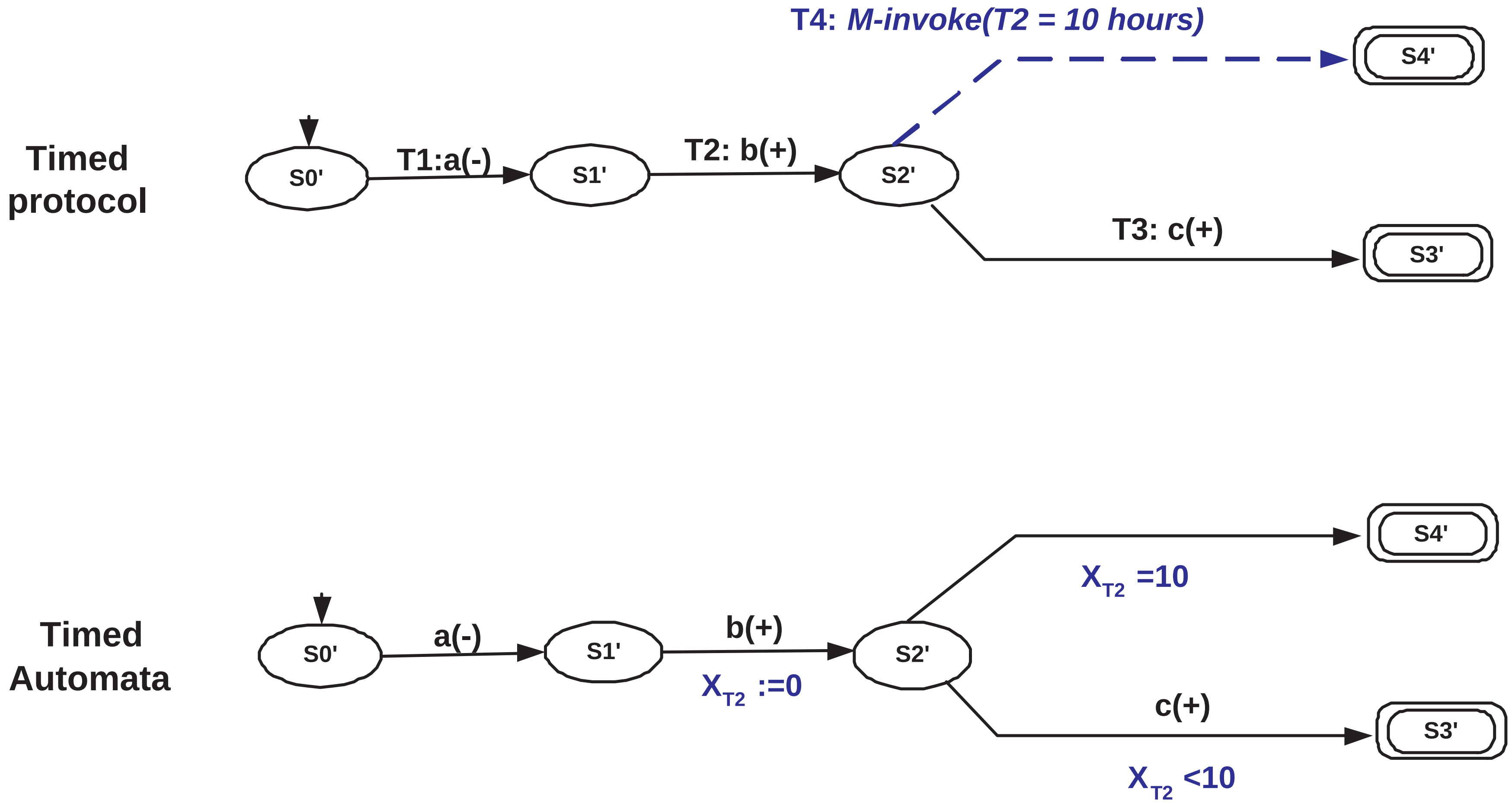
Timed protocols: decision problems

Strong link with the theory of timed automata

- Formal notation to model behavior of real time systems
- State-transition graphs with timing contraints using real-valued clock variables
- Extensively studied formalism

Mapping: timed protocols → timed automata

Mapping to timed automata



Main results

A few lessons from Timed Automata

- Closed under all boolean operations but complementation
- Silent transitions strictly increase expressiveness when they reset clocks

Timed protocols = New subclass of timed automata

→ closed under complementation

Agenda

- Web services vision and technologies
- Representing web service protocols
- Analysis and management of web service protocols
- **Summary and outlook**

Summary and outlook

- Analysis and management of web service protocols

  So far, focus on primitives for facilitating automation of services development and interoperability

  - Extension to business protocols augmented with transactional abstractions
  - Trust negotiation and security protocols in Web services
  - Analysis of multi-party protocols (consistency analysis)
  - Composite services analysis and synthesis
  - Protocol discovery

Thanks