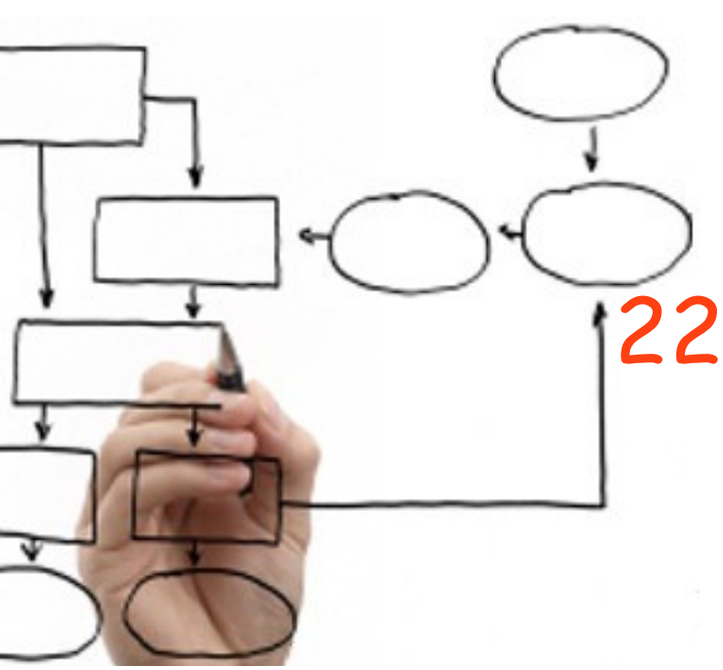
Methods for the specification and verification of business processes

MPB (6 cfu, 295AA)

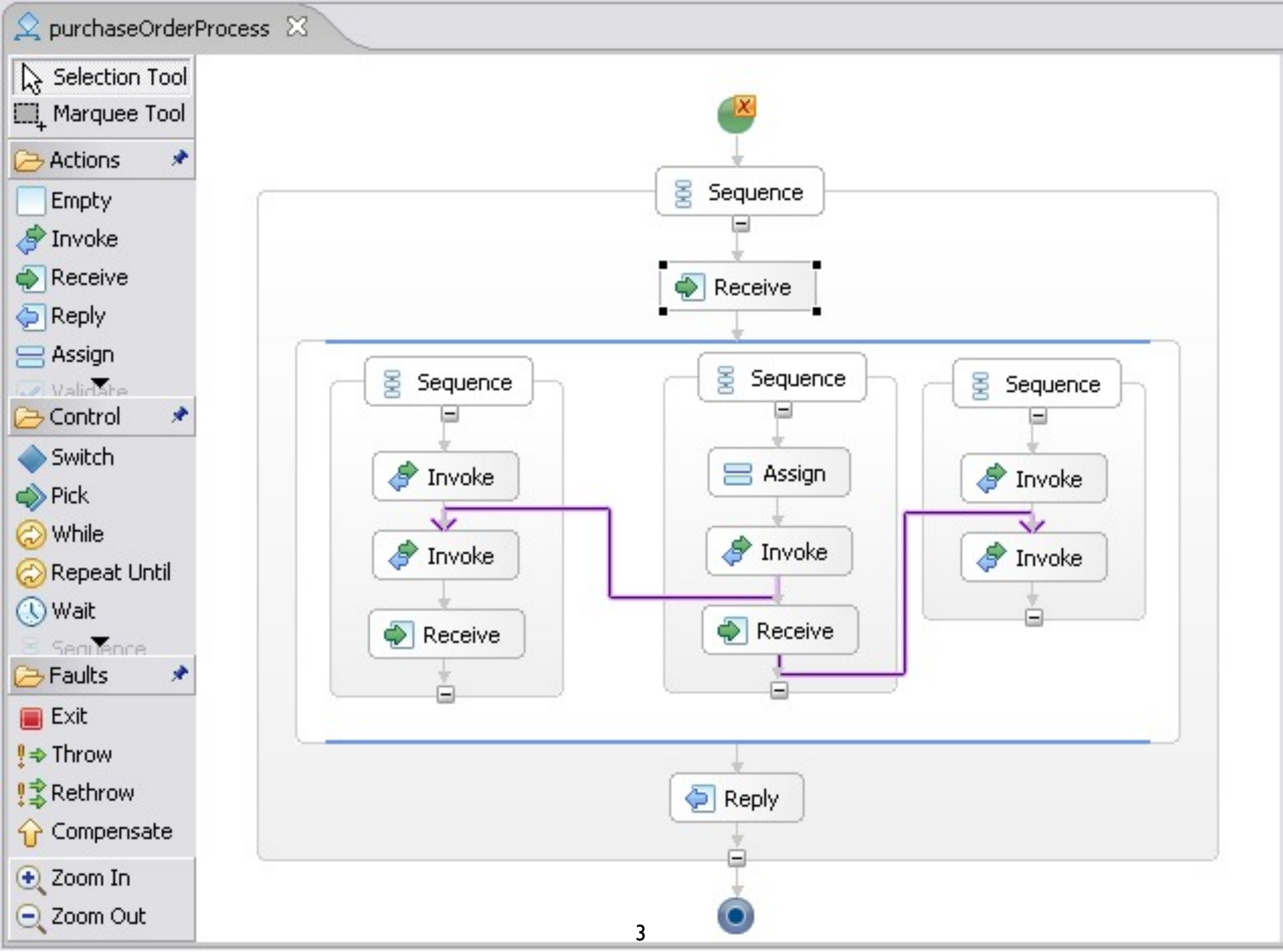
Roberto Bruni
http://www.di.unipi.it/~bruni



22 - Business process execution language

Object

We overview the key features of BPEL


purchaseOrderProcess
Selection Tool
Marquee Tool
Actions
Empty
Invoke
Receive
Reply
Assign
Control
Switch
Pick
While
Repeat Until
Wait
Faults
Exit
Throw
Rethrow
Compensate
Zoom In
Zoom Out
Sequence
Receive
Sequence
Invoke
Invoke
Receive
Sequence
Assign
Invoke
Receive
Sequence
Invoke
Invoke
Reply

BPEL

Business process execution language

Also known as:
Web Services Business Process Execution Language (WS-BPEL)
Business Process Execution Language for Web Services (BPEL4WS)

it is a standard executable language for orchestrating the use of Web Service within business processes

it deals with import / export information, remote invocation, correlation, fault handling, compensation

Web services

Web services fix a standard for interoperability between heterogeneous, loosely coupled, remote software applications
(separately developed, running on different platforms)
over (not only) the HTTP protocol

Informally:
web services are for software
what web sites are for human

WS basics

Services must be made available on the web
(need a server)

Services must be advertised over the web
(need some repositories)

Service repositories must be queried
(need service description)

Services must be invoked
(need standard communication format)

XMLification

WSFL, BPEL, ...

UDDI

WSDL

SOAP

HTTP, HTTPS, SMTP

Service composition

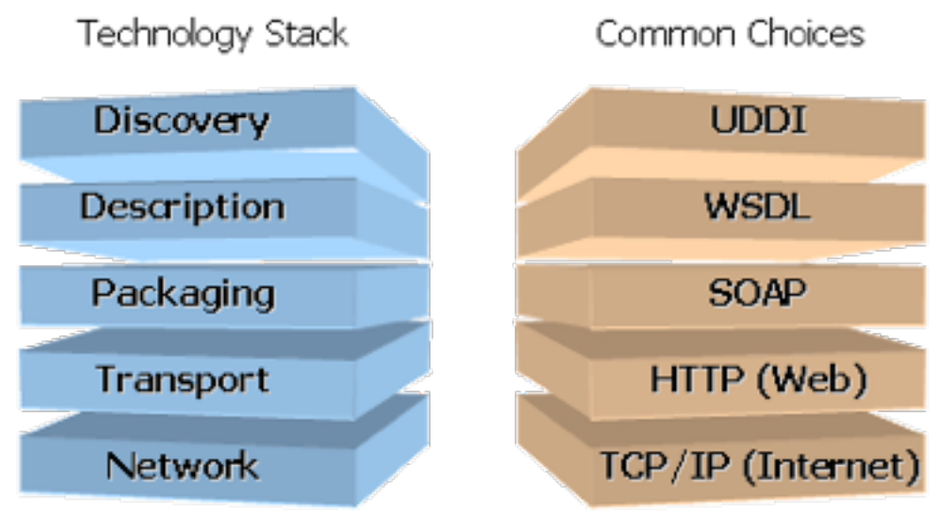
Service discovery

Service publication

Service description

XML based messaging

Network



WS-*


Management
Choreography – CDL4WS
Orchestration – BPEL4WS
Business processes
WS-Reliability
WS-Security
Transactions
Coordination
Context
Quality of service
UDDI
Discovery
WSDL
Description
SOAP
XML
Message
HTTP, JMS, SMTP
Transport

Web Services Standards Overview

Interoperability Issues

Basic Profile 1.1 WS-I Final Specification

▲ **Basic Profile** – The Basic Profile 1.1 provides implementation guidelines for how related set of non-proprietary Web Service specifications should be used together for best interoperability.

Basic Profile 1.2 WS-I Working Group Draft

▲ **Basic Profile** – The Basic Profile 1.2 builds on Basic Profile 1.1 by incorporating Basic Profile 1.1 errata, requirements from Simple SOAP Binding Profile 1.0, and adding support for WS-Addressing and MTOM.

Basic Profile 2.0 WS-I Working Group Draft

▲ **Basic Profile** – The Basic Profile 2.0 is an update of WS-I BP that includes a profile of SOAP 1.2.

Attachments Profile 1.0 WS-I Final Specification

▲ **Attachments Profile** – The Attachment Profile 1.0 complements the Basic Profile 1.1 to add support for interoperable SOAP Messages with attachments-based Web Services.

Simple SOAP Binding Profile 1.0 WS-I Final Specification

▲ **Simple SOAP Binding Profile** – The Simple SOAP Binding Profile consists of those Basic Profile 1.0 requirements related to the serialization of the envelope and its representation in the message.

Basic Security Profile 1.0 WS-I Board Approval Draft

▲ **Basic Security Profile** defines the WS-I Basic Security Profile 1.0, based on a set of non-proprietary Web services specifications, along with clarifications and amendments to those specifications which promote interoperability.

REL Token Profile 1.0 WS-I Working Group Draft

▲ **REL Token Profile** is based on a non-proprietary Web services specification, along with clarifications and amendments to that specification which promote interoperability.

SAML Token Profile 1.0 WS-I Working Group Draft

▲ **SAML Token Profile** is based on a non-proprietary Web services specification, along with clarifications and amendments to that specification which promote interoperability.

Conformance Claim Attachment Mechanism (CCAM) 1.0 WS-I Final Specification

▲ **Conformance Claim Attachment Mechanism (CCAM)** catalogues mechanisms that can be used to attach WS-I Profile Conformance Claims to Web services artefacts (e.g., WSDL descriptions, UDDI registries).

Reliable Asynchronous Messaging Profile (RAMP) 1.0 WS-I Working Draft

▲ **Reliable Asynchronous Messaging Profile (RAMP)** is a profile, in the fashion of the WS-I profiles, that enables, among other things, basic B2B integration scenarios using Web services technologies.

Business Process Specifications

Business Process Execution Language for Web Services 1.1 (BPEL4WS) · 1.1 · BEA Systems, IBM, Microsoft, SAP, Siebel Systems · OASIS-Standard

▲ **Business Process Execution Language for Web Services 1.1** (BPEL4WS) provides a language for the formal specification of business processes and business interaction protocols using Web Services.

WS-Choreography Model Overview 1.0 W3C Working Draft

▲ **WS-Choreography Model Overview** defines the format and structure of the (SOAP) messages that are exchanged, and the sequence and conditions in which the messages are exchanged.

Web Service Choreography Interface (WSCI) · 1.0 · W3C Sun Microsystems, SAP, BEA Systems and Intalio · Note

▲ **Web Service Choreography Interface** (WSCI) describes how Web Service operations can be choreographed in the context of a message exchange in which the Web Service participates.

Web Service Choreography Description Language (CDL4WS) · 1.0 · W3C Candidate Recommendation

▲ **Web Service Choreography Description Language** (CDL4WS) is to specify a declarative, XML based language that defines from a global viewpoint the common and complementary observable behaviour, where message exchanges occur, and when the jointly agreed ordering rules are satisfied.

Business Process Execution Language for Web Services 2.0 (BPEL4WS) · 2.0 OASIS, BEA Systems, IBM, Microsoft, SAP, Siebel Systems · Committee Draft

▲ **Business Process Execution Language for Web Services 2.0** (BPEL4WS) provides a language for the formal specification of business processes and business interaction protocols using Web Services.

Business Process Management Language (BPML) 1.1 BPMI.org Final Draft

▲ **Business Process Management Language** (BPML) provides a meta-language for expressing business processes and supporting entities.

XML Process Definition Language (XPDL) 2.0 Final

▲ **XML Process Definition Language** (XPDL) provides an XML file format that can be used to interchange process models between tools.

Management Specifications

Management Using Web Services (WSDM-MUWS) 1.0 OASIS OASIS-Standard

▲ **Web Service Distributed Management:** Management Using Web Services (WSDM-MUWS) defines how an IT resource connected to a network provides manageability interfaces such that the IT resource can be managed locally and from remote locations using Web services technologies.

Management Of Web Services (WSDM-MOWS) 1.0 OASIS OASIS-Standard

▲ **Web Service Distributed Management:** Management Of Web Services (WSDM-MOWS) addresses management of the components that form the network, the Web services endpoints, using Web services protocols.

WS-Management AMD, Dell, Intel, Microsoft and Sun Microsystems Published Specification

▲ **WS-Management** describes a general SOAP-based protocol for managing systems such as PCs, servers, devices, Web services and other applications, and other manageable entities.

Service Modeling Language IBM, BEA, BMC, Cisco, Dell, HP, Intel, Microsoft, Sun Draft Specification

▲ **Service Modeling Language** (SML) is used to model complex IT services and systems, including their structure, constraints, policies, and best practices.

Presentation Specifications

Web Services for Remote Portlets (WSRP) 2.0 OASIS Committee Draft

▶ **Web Services for Remote Portlets** (WSRP) defines a set of interfaces and related semantics which standardize interactions with components providing user-facing markup, including the processing of user interactions with that markup.

Metadata Specifications

WS-Policy 1.5 W3C Working Draft

▲ **WS-Policy** describes the capabilities and constraints of the policies on intermediaries and endpoints (e.g. business rules, required security tokens, supported encryption algorithms, privacy rules).

WS-PolicyAssertions 1.1 BEA Systems, IBM, Microsoft, SAP Public Draft

▲ **WS-PolicyAssertions** provides an initial set of assertions to address some common needs of Web Services applications.

WS-PolicyAttachment 1.2 W3C W3C Member Submission

▲ **WS-PolicyAttachment** defines two general-purpose mechanisms for associating policies with the subjects to which they apply; the policies may be defined as part of existing metadata about the subject or the policies may be defined independently and associated through an external binding to the subject.

WS-Discovery Microsoft, BEA Systems, Canon, Intel and webMethods Draft

▲ **WS-Discovery** defines a multicast discovery protocol for dynamic discovery of services on ad-hoc and managed networks.

WS-MetadataExchange 1.1 BEA Systems, Computer Associates, IBM, Microsoft, SAP, Sun Microsystems and webMethods Public Draft

▲ **WS-MetadataExchange** enables a service to provide metadata to others through a Web services interface. Given only a reference to a Web service, a user can access a set of WSDL/SOAP operations to retrieve the metadata that describes the service.

Universal Description, Discovery and Integration (UDDI) 3.0.2 OASIS OASIS-Standard

▲ **Universal Description, Discovery and Integration (UDDI)** defines a set of services supporting the description and discovery of businesses, organizations, and other Web services providers, the Web services they make available, and the technical interfaces which may be used to access those services.

Web Service Description Language 2.0 SOAP Binding 2.0 W3C · Working Draft

▲ **Web Service Description Language SOAP Binding** describes the concrete details for using WSDL 2.0 in conjunction with SOAP 1.1 protocol.

Web Service Description Language 2.0 Core 2.0 W3C Candidate Recommendation

▲ **Web Service Description Language 2.0 Core** is an XML-based language for describing Web services and how to access them. It specifies the location of the service and the operations (or methods) the service exposes.

Web Service Description Language 1.1 1.1 W3C Note

▲ **Web Service Description Language 1.1** is an XML-based language for describing Web services and how to access them. It specifies the location of the service and the operations (or methods) the service exposes.

Reliability Specifications

WS-ReliableMessaging 1.1 OASIS Committee Draft

▲ **WS-ReliableMessaging** describes a protocol that allows Web services to communicate reliable in the presence of software component, system, or network failures. It defines a SOAP binding that is required for interoperability.

WS-Reliable Messaging Policy Assertion (WS-RM Policy) 1.1 OASIS Committee Draft

▲ **Web Services ReliableMessaging Policy Assertion** (WS-RM Policy) describes a domain-specific policy assertion for WS-ReliableMessaging that that can be specified within a policy alternative as defined in WS-Policy Framework.

WS-Reliability 1.1 OASIS OASIS-Standard

▲ **WS-Reliability** is a SOAP-based protocol for exchanging SOAP messages with guaranteed delivery, no duplicates, and guaranteed message ordering. WS-Reliability is defined as SOAP header extensions and is independent of the underlying protocol. This specification contains a binding to HTTP.

Security Specifications

WS-Security 1.1 OASIS OASIS-Standard

▲ **WS-Security** is a communications protocol providing a means for applying security to Web Services.

WS-SecurityPolicy 1.1 IBM, Microsoft, RSA Security, VeriSign Public Draft

▲ **WS-SecurityPolicy** defines how to describe policies related to various features defined in the WS-Security specification.

WS-Security: SOAP Message Security 1.1 OASIS Public Review Draft

▲ **WS-Security: SOAP Message Security** describes enhancements to SOAP messaging to provide message integrity and confidentiality. Specifically, this specification provides support for multiple security token formats, trust domains, signature formats and encryption technologies. The token formats and semantics for using these are defined in the associated profile documents.

WS-Security: Username Token Profile 1.1 OASIS Public Review Draft

▲ **WS-Security: Username Token Profile** describes how a Web Service consumer can supply a Username Token as a means of identifying the requestor by username, and optionally using a password (or shared secret, etc.) to authenticate that identity to the Web Service producer.

WS-Security: Kerberos Binding 1.0 Microsoft, IBM, OASIS Working Draft

▲ **WS-Security: Kerberos Binding** defines how to encode Kerberos tickets and attach them to SOAP messages. As well, it specifies how to add signatures and encryption to the SOAP message, in accordance with WS-Security, which uses and references the Kerberos tokens.

WS-Federation 1.0 IBM, Microsoft, BEA Systems, RSA Security, and VeriSign Initial Draft

▲ **WS-Federation** describes how to manage and broker the trust relationships in a heterogeneous federated environment including support for federated identities.

WS-Security: SAML Token Profile 1.1 OASIS Public Review Draft

▲ **WS-Security: SAML Token Profile** defines the use of Security Assertion Markup Language (SAML) v1.1 assertions in the context of WSS: SOAP Message Security including for the purpose of securing SOAP messages and SOAP message exchanges.

WS-Trust BEA Systems, Computer Associates, IBM, Layer 7 Technologies, Microsoft, Netegrity, Oblix, OpenNetwork, Ping Identity Corporation, Reactivity, RSA Security, VeriSign and Westbridge Technology · Initial Draft

▲ **WS-Trust** describes a framework for trust models that enables Web Services to securely interoperate. It uses WS-Security base mechanisms and defines additional primitives and extensions for security token exchange to enable the issuance and dissemination of credentials within different trust domains.

WS-Security: X.509 Certificate Token Profile 1.1 OASIS Public Review Draft

▲ **WS-Security: X.509 Certificate Token Profile** describes the use of the X.509 authentication framework with the WS-Security: SOAP Message Security specification.

WS-SecureConversation BEA Systems, Computer Associates, IBM, Layer 7 Technologies, Microsoft, Netegrity, Oblix, OpenNetwork, Ping Identity Corporation, Reactivity, RSA Security, VeriSign and Westbridge Technology · Public Draft

▲ **WS-SecureConversation** specifies how to manage and authenticate message exchanges between parties including security context exchange and establishing and deriving session keys.

Transaction Specifications

WS-Coordination 1.1 OASIS Working Draft

▲ **WS-Coordination** describes an extensible framework for providing protocols that coordinate the actions of distributed applications.

WS-Business Activity 1.1 OASIS Working Draft

▲ **WS-Business Activity** provides the definition of the business activity coordination type that is to be used with the extensible coordination framework described in the WS-Coordination specification.

WS-Atomic Transaction 1.1 OASIS Committee Draft

▲ **WS-Atomic Transaction** defines protocols that enable existing transaction processing systems to wrap their proprietary protocols and interoperate across different hardware and software vendors.

WS-Composite Application Framework (WS-CAF) 1.0 · Arjuna Technologies, Fujitsu, IONA, Oracle and Sun Microsystems Committee Specification

▲ **WS-Composite Application Framework** (WS-CAF) is a collection of three specifications aimed at solving problems that arise when multiple Web Services are used in combination. It proposes standard, interoperable mechanisms for managing shared context and ensuring business processes achieve predictable results and recovery from failure.

WS-Context (WS-CTX) 1.0 Arjuna Technologies, Fujitsu, IONA, Oracle and Sun Microsystems Committee Draft

▲ **WS-Context** (WS-CTX) is intended as a lightweight mechanism for allowing multiple Web Services to share a common context.

WS-Coordination Framework (WS-CF) 1.0 · Arjuna Technologies, Fujitsu, IONA, Oracle and Sun Microsystems Committee Draft

▲ **WS-Coordination Framework** (WS-CF) allows the management and coordination in a Web Services interaction of a number of activities related to an overall application.

WS-Transaction Management (WS-TXM) 1.0 · Arjuna Technologies, Fujitsu, IONA, Oracle and Sun Microsystems Committee Draft

▲ **WS-Transaction Management** (WS-TXM) defines a core infrastructure service consisting of a Transaction Service for Web Services.

Resource Specifications

Web Services Resource Framework (WSRF) 1.2 OASIS OASIS-Standard

▲ **Web Services Resource Framework** (WSRF) defines a family of specifications for accessing stateful resources using Web Services.

WS-BaseFaults (WSRF) 1.2 OASIS Working Draft

▲ **WS-BaseFaults** (WSRF) defines a base set of information that may appear in fault messages. WS-BaseFaults defines an XML schema type for base faults, along with rules for how this base fault type is used and extended by Web Services.

WS-ServiceGroup (WSRF) 1.2 OASIS Working Draft

▲ **WS-ServiceGroup** (WSRF) defines a means by which Web Services and WS-Resources can be aggregated or grouped together for a domain specific purpose.

WS-ResourceProperties 1.2 OASIS Working Draft

▲ **WS-ResourceProperties** specifies the means by which the definition of the properties of a WS-Resource may be declared as part of the Web Service interface. The declaration of the WS-Resource properties represents a projection of or a view on the WS-Resource state.

WS-ResourceLifetime 1.2 OASIS Working Draft

▲ **WS-ResourceLifetime** is to standardize the terminology, concepts, message exchanges, WSDL and XML needed to monitor the lifetime of, and destroy WS-Resources. Additionally, it defines resource properties that may be used to inspect and monitor the lifetime of a WS-Resource.

WS-Transfer W3C W3C Member Submission

▲ **WS-Transfer** describes a general SOAP-based protocol for accessing XML representations of Web service-based resources.

Resource Representation SOAP Header Block (RRSHB) W3C Recommendation

▲ **Resource Representation SOAP Header Block** (RRSHB) complements MTOM by defining mechanisms for describing and conveying non-XML resource representations in a SOAP 1.2 message.

Messaging Specifications

WS-Notification 1.3 OASIS OASIS-Standard

▶ **WS-Notification** is a family of related white papers and specifications that define a standard Web services approach to notification using a topic-based publish/subscribe pattern.

WS-BrokeredNotification 1.3 OASIS OASIS-Standard

▶ **WS-BrokeredNotification** defines the interface for the NotificationBroker. A NotificationBroker is an intermediary, which, among other things, allows publication of messages from entities that are not themselves service providers.

WS-BaseNotification 1.3 OASIS OASIS-Standard

▶ **WS-BaseNotification** standardizes the terminology, concepts, operations, WSDL and XML needed to express the basic roles involved in Web services publish and subscribe for notification message exchange.

WS-Eventing W3C Public Draft

▶ **WS-Eventing** defines a baseline set of operations that allow Web services to provide asynchronous notifications to interested parties.

WS-Addressing – Core 1.0 W3C Recommendation

▶ **WS-Addressing – Core** provides transport-neutral mechanisms to address Web services and messages. This specification defines XML elements to identify Web service endpoints and to secure end-to-end endpoint identification in messages.

WS-Enumeration Systinet, Microsoft, Sonic Software, BEA Systems and Computer Associates Public Draft

▶ **WS-Enumeration** describes a general SOAP-based protocol for enumerating a sequence of XML elements that is suitable for traversing logs, message queues, or other linear information models.

WS-Topics 1.3 OASIS OASIS-Standard

▶ **WS-Topics** defines three topic expression dialects that can be used as sub-scription expressions in subscribe request messages and other parts of the WS-Notification system.

WS-Addressing – WSDL Binding 1.0 W3C Candidate Recommendation

▶ **WS-Addressing – WSDL Binding** defines how the abstract properties defined in Web Services Addressing – Core are described using WSDL.

WS-Addressing – SOAP Binding 1.0 W3C Recommendation

▶ **WS-Addressing – SOAP Binding** provides transport-neutral mechanisms to address Web services and messages.

SOAP

SOAP 1.2 W3C Recommendation

SOAP Message Transmission Optimization Mechanism (MTOM) 1.0 · W3C Recommendation

▶ **SOAP Message Transmission Optimization Mechanism** describes an abstract feature for optimizing the transmission and/or wire format of a SOAP message.

SOAP 1.1 W3C Note

▶ **SOAP** is a lightweight, XML-based protocol for exchange of information in a decentralized, distributed environment.

XML Specifications

XML 1.1 1.1 W3C Recommendation

▶ **XML** – Extensible Markup Language is a pared-down version of SGML, designed especially for Web documents. It allows one to create own customized tags, enabling the definition, transmission, validation, and interpretation of data between applications and between organizations.

XML 1.0 1.0 W3C Recommendation

▶ **XML** – Extensible Markup Language is a pared-down version of SGML, designed especially for Web documents. It allows one to create own customized tags, enabling the definition, transmission, validation, and interpretation of data between applications and between organizations.

Namespaces in XML 1.1 W3C Recommendation

▶ **Namespaces in XML** provides a simple method for qualifying element and attribute names used in XML documents by associating them with namespaces identified by IRI references.

XML Information Set 1.0 W3C Recommendation

▶ **XML Information Set** is an abstract data set to provide a consistent set of definitions for use in other specifications that need to refer to the information in a well-formed XML document.

XML Schema 1.1 W3C Working Draft

▶ **XML Schema** – XML Schema Definition Language is an XML language for describing and constraining the content of XML documents.

XML binary Optimized Packaging (XOP) 1.0 W3C Recommendation

▶ **XML binary Optimized Packaging (XOP)** is an XML language for describing and constraining the content of XML documents.

Describing Media Content of Binary Data in XML (DMCBDX) W3C Note

▶ **Describing Media Content of Binary Data in XML (DMCBDX)** specifies how to indicate the content-type associated with binary element content in an XML document and to specify, in XML Schema, the expected content-type(s) associated with binary element content.

Standards Bodies

OASIS The **Organization for the Advancement of Structured Information Standards (OASIS)** is a not-for-profit, international consortium that drives the development, convergence, and adoption of e-business standards. The consortium produces more Web services standards than any other organization along with standards for security, e-business, and standardization efforts in the public sector and for application-specific markets. Founded in 1993, OASIS has more than 4,000 participants representing over 600 organizations and individual members in 100 countries.

W3C The **World Wide Web Consortium (W3C)** was created in October 1994 to lead the World Wide Web to its full potential by developing common protocols that promote its evolution and ensure its interoperability. W3C has over 350 Member organizations from all over the world and has earned international recognition for its contributions to the growth of the Web. W3C is designing the infrastructure, and defining the architecture and the core technologies for Web services. In September 2000, W3C started the XML Protocol Activity to address the need for an XML-based protocol for application-to-application messaging. In January 2002, the Web Services Activity was launched, subsuming the XML Protocol Activity and extending its scope.

WS-I The **Web Services Interoperability Organization (WS-I)** is an open industry organization chartered to promote Web services interoperability across platforms, operating systems and programming languages. The organization's diverse community of Web services leaders helps customers to develop interoperable Web services by providing guidance, recommended practices and supporting resources. Specifically, WS-I creates, promotes and supports generic protocols for the interoperable exchange of messages between Web services.

IETF The **Internet Engineering Task Force (IETF)** is a large open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet.

Dependencies

Messaging Specifications

SOAP 1.1
SOAP 1.2
SOAP Message Transmission Optimization Mechanism
WS-Notification
WS-BaseNotification
WS-Topics
WS-BrokeredNotification
WS-Addressing – Core
WS-Addressing – SOAP Binding
WS-Eventing
WS-Enumeration

Resource | Security | Metadata

Metadata Specifications

WS-Policy
WS-PolicyAssertions
WS-PolicyAttachment
WS-Discovery
WS-MetadataExchange
Universal Description, Discovery and Integration
Web Service Description Language 1.1
Web Service Description Language 2.0 Core
Web Service Description Language 2.0 SOAP Binding

Security | Messaging

Security Specifications

WS-Security
WS-Security: SOAP Message Security
WS-Security: Kerberos Binding
WS-Security: SAML Token Profile
WS-Security: X.509 Certificate Token Profile
WS-Security: Username Token Profile
WS-SecurityPolicy
WS-Trust
WS-Federation
WS-SecureConversation

Reliability | Messaging | Metadata

Reliability Specifications

WS-ReliableMessaging
WS-Reliability
WS-Reliable Messaging Policy Assertion

Transaction | Basic Profile | Security | Metadata

Resource Specifications

Web Service Resource Framework
WS-BaseFaults
WS-ServiceGroup
WS-ResourceProperties
WS-ResourceLifetime
WS-Transfer
Resource Representation SOAP Header Block (RRSHB)

Transaction | Security | Messaging

Management Specifications

WS-Management
Management Of Web Services
Management Using Web Services
Service Modeling Language

Resource | Meta | Security | Messaging

Business Process Specifications

Business Process Execution Language for Web Services
Web Service Choreography Description Language
Web Service Choreography Interface
WS-Choreography Model Overview
Business Process Management Language
Business Process Execution Language for Web Serv. 2.0
XML Process Definition Language

Reliability | Security | Messaging | Transaction

Transaction Specifications

WS-Business Activity
WS-Atomic Transaction
WS-Coordination
WS-Composite Application Framework
WS-Transaction Management
WS-Context
WS-Coordination Framework

Metadata | Messaging | Reliability | Security

Presentation Specifications

Web Services for Remote Portlets

Reliab. | Secur. | Mess.

innoQ

innoQ Deutschland GmbH
Halskestraße 17
D-40880 Ratingen
Phone +49 2102 77162-100
info@innoq.com · www.innoq.com

innoQ Schweiz GmbH
Gewerbestrasse 11
CH-6330 Cham
Phone +41 41 743 01 11

Version 3.0 · February 2007

Birth of BPEL

IBM was pushing for a standard called WSFL

Microsoft was pushing for a technology called XLANG

Intalio was pushing for BPML

IBM and Microsoft merged their efforts and pushed together for BPEL (a hybrid WSFL+XLANG) and BPEL was soon widely adopted

Life of BPEL

BPEL4WS 1.0 (2002) by BEA, IBM, Microsoft

SAP + Siebel joined the effort
BPEL 1.1 (2003)
submitted to OASIS

Adobe + HP + NEC + Oracle + Sun + many more joined
WS-BPEL 2.0 (2005)

The problem with BPEL

BPEL is not a graphical language

BPEL is an XML dialect

Machines like XML
Humans being should not like XML

A typical BPEL tutorial

Turn to page 4 of any BPEL tutorial
(the first couple of pages are just a verbal introduction)
and you find the first small example...

... of about two pages of formatted XML code
(with all actual namespaces
to avoid any misunderstanding)

```
<process name="purchaseOrderProcess" targetNamespace="http://example.com/ws-bp/purchase" xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable" xmlns:lns="http://manufacturing.org/wsdl/purchase">

   <documentation xml:lang="EN">
      A simple example of a WS-BPEL process for handling a purchase order.
   </documentation>

   <partnerLinks>
      <partnerLink name="purchasing" partnerLinkType="lns:purchasingLT" myRole="purchaseService" />
      <partnerLink name="invoicing" partnerLinkType="lns:invoicingLT" myRole="invoiceRequester" partnerRole="invoiceService" />
      <partnerLink name="shipping" partnerLinkType="lns:shippingLT" myRole="shippingRequester" partnerRole="shippingService" />
      <partnerLink name="scheduling" partnerLinkType="lns:schedulingLT" partnerRole="schedulingService" />
   </partnerLinks>

   <variables>
      <variable name="PO" messageType="lns:POMessage" />
      <variable name="Invoice" messageType="lns:InvMessage" />
      <variable name="shippingRequest" messageType="lns:shippingRequestMessage" />
      <variable name="shippingInfo" messageType="lns:shippingInfoMessage" />
      <variable name="shippingSchedule" messageType="lns:scheduleMessage" />
   </variables>

   <faultHandlers>
      <catch faultName="lns:cannotCompleteOrder" faultVariable="POFault" faultMessageType="lns:orderFaultType">
         <reply partnerLink="purchasing" portType="lns:purchaseOrderPT" operation="sendPurchaseOrder" variable="POFault" faultName="cannotCompleteOrder" />
      </catch>
   </faultHandlers>

   <sequence>
      <receive partnerLink="purchasing" portType="lns:purchaseOrderPT" operation="sendPurchaseOrder" variable="PO" createInstance="yes">
         <documentation>Receive Purchase Order</documentation>
      </receive>

      <flow>
         <documentation>
            A parallel flow to handle shipping, invoicing and scheduling
         </documentation>
         <links>
            <link name="ship-to-invoice" />
            <link name="ship-to-scheduling" />
         </links>
         <sequence>
            <assign>
               <copy>
                  <from>$PO.customerInfo</from>
                  <to>$shippingRequest.customerInfo</to>
               </copy>
            </assign>
            <invoke partnerLink="shipping" portType="lns:shippingPT" operation="requestShipping" inputVariable="shippingRequest" outputVariable="shippingInfo">
               <documentation>Decide On Shipper</documentation>
               <sources>
                  <source linkName="ship-to-invoice" />
               </sources>
            </invoke>
            <receive partnerLink="shipping" portType="lns:shippingCallbackPT" operation="sendSchedule" variable="shippingSchedule">
               <documentation>Arrange Logistics</documentation>
               <sources>
                  <source linkName="ship-to-scheduling" />
               </sources>
            </receive>
         </sequence>
         <sequence>
            <invoke partnerLink="invoicing" portType="lns:computePricePT" operation="initiatePriceCalculation" inputVariable="PO">
               <documentation>
                  Initial Price Calculation
               </documentation>
            </invoke>
            <invoke partnerLink="invoicing" portType="lns:computePricePT" operation="sendShippingPrice" inputVariable="shippingInfo">
               <documentation>
                  Complete Price Calculation
               </documentation>
               <targets>
                  <target linkName="ship-to-invoice" />
               </targets>
            </invoke>
            <receive partnerLink="invoicing" portType="lns:invoiceCallbackPT" operation="sendInvoice" variable="Invoice" />
         </sequence>
         <sequence>
            <invoke partnerLink="scheduling" portType="lns:schedulingPT" operation="requestProductionScheduling" inputVariable="PO">
               <documentation>
                  Initiate Production Scheduling
               </documentation>
            </invoke>
            <invoke partnerLink="scheduling" portType="lns:schedulingPT" operation="sendShippingSchedule" inputVariable="shippingSchedule">
               <documentation>
                  Complete Production Scheduling
               </documentation>
               <targets>
                  <target linkName="ship-to-scheduling" />
               </targets>
            </invoke>
         </sequence>
      </flow>
      <reply partnerLink="purchasing" portType="lns:purchaseOrderPT" operation="sendPurchaseOrder" variable="Invoice">
         <documentation>Invoice Processing</documentation>
      </reply>
   </sequence>
</process>
```

A syntax called semantics

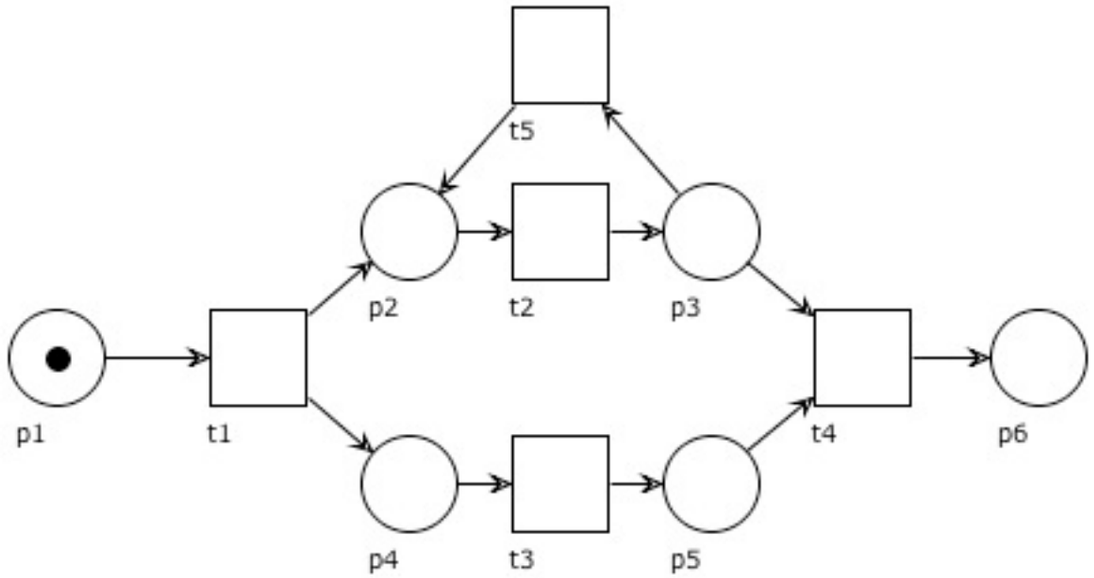
Learning BPEL by looking at XML documents

is like

learning Petri nets by looking at PNML documents

or similar to

learning Java by looking at the bytecode



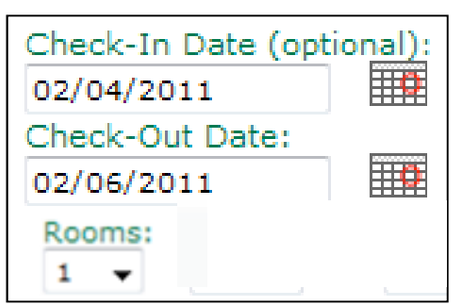

```xml
<?xml version="1.0" encoding="UTF-8"?><pnml>
  <net type="http://www.informatik.hu-berlin.de/top/pntd/ptNetb" id="noID">
    <place id="p6">
      <name>
        <text>p6</text>
        <graphics>
          <offset x="430" y="270"/>
        </graphics>
      </name>
      <graphics>
        <position x="430" y="230"/>
        <dimension x="40" y="40"/>
      </graphics>
    </place>
    <place id="p5">
      <name>
        <text>p5</text>
        <graphics>
          <offset x="300" y="320"/>
        </graphics>
      </name>
      <graphics>
        <position x="300" y="280"/>
        <dimension x="40" y="40"/>
      </graphics>
    </place>
    <place id="p4">
      <name>
        <text>p4</text>
        <graphics>
          <offset x="180" y="320"/>
        </graphics>
      </name>
      <graphics>
        <position x="180" y="280"/>
        <dimension x="40" y="40"/>
      </graphics>
    </place>
    <place id="p3">
      <name>
        <text>p3</text>
        <graphics>
          <offset x="300" y="220"/>
        </graphics>
      </name>
      <graphics>
        <position x="300" y="180"/>
        <dimension x="40" y="40"/>
      </graphics>
    </place>
    <place id="p2">
      <name>
        <text>p2</text>
        <graphics>
          <offset x="180" y="220"/>
        </graphics>
      </name>
      <graphics>
        <position x="180" y="180"/>
        <dimension x="40" y="40"/>
      </graphics>
    </place>
    <place id="p1">
      <name>
        <text>p1</text>
        <graphics>
          <offset x="40" y="270"/>
        </graphics>
      </name>
      <graphics>
        <position x="40" y="230"/>
        <dimension x="40" y="40"/>
      </graphics>
      <initialMarking>
        <text>1</text>
      </initialMarking>
    </place>
    <transition id="t3">
      <name>
        <text>t3</text>
        <graphics>
          <offset x="240" y="320"/>
        </graphics>
      </name>
      <graphics>
        <position x="240" y="280"/>
        <dimension x="40" y="40"/>
      </graphics>
      <toolspecific tool="WoPeD" version="1.0">
        <time>0</time>
        <timeUnit>1</timeUnit>
        <orientation>1</orientation>
      </toolspecific>
    </transition>
    <transition id="t2">
      <name>
        <text>t2</text>
        <graphics>
          <offset x="240" y="220"/>
        </graphics>
      </name>
      <graphics>
        <position x="240" y="180"/>
        <dimension x="40" y="40"/>
      </graphics>
      <toolspecific tool="WoPeD" version="1.0">
        <time>0</time>
        <timeUnit>1</timeUnit>
        <orientation>1</orientation>
      </toolspecific>
    </transition>
    <transition id="t1">
      <name>
        <text>t1</text>
        <graphics>
          <offset x="120" y="270"/>
        </graphics>
      </name>
      <graphics>
        <position x="120" y="230"/>
        <dimension x="40" y="40"/>
      </graphics>
      <toolspecific tool="WoPeD" version="1.0">
        <time>0</time>
        <timeUnit>1</timeUnit>
        <orientation>1</orientation>
      </toolspecific>
    </transition>
    <transition id="t4">
      <name>
        <text>t4</text>
        <graphics>
          <offset x="360" y="270"/>
        </graphics>
      </name>
      <graphics>
        <position x="360" y="230"/>
        <dimension x="40" y="40"/>
      </graphics>
      <toolspecific tool="WoPeD" version="1.0">
        <time>0</time>
        <timeUnit>1</timeUnit>
        <orientation>1</orientation>
      </toolspecific>
    </transition>
    <transition id="t5">
      <name>
        <text>t5</text>
        <graphics>
          <offset x="240" y="150"/>
        </graphics>
      </name>
      <graphics>
        <position x="240" y="110"/>
        <dimension x="40" y="40"/>
      </graphics>
      <toolspecific tool="WoPeD" version="1.0">
        <time>0</time>
        <timeUnit>1</timeUnit>
        <orientation>1</orientation>
      </toolspecific>
    </transition>
    <arc id="a9" source="t1" target="p4">
      <inscription>
        <text>1</text>
      </inscription>
      <graphics/>
      <toolspecific tool="WoPeD" version="1.0">
        <probability>1.0</probability>
        <displayProbabilityOn>false</displayProbabilityOn>
        <displayProbabilityPosition x="500.0" y="0.0"/>
      </toolspecific>
    </arc>
    <arc id="a17" source="p3" target="t4">
      <inscription>
        <text>1</text>
      </inscription>
      <graphics/>
      <toolspecific tool="WoPeD" version="1.0">
        <probability>1.0</probability>
        <displayProbabilityOn>false</displayProbabilityOn>
        <displayProbabilityPosition x="500.0" y="0.0"/>
      </toolspecific>
    </arc>
    <arc id="a14" source="p5" target="t4">
      <inscription>
        <text>1</text>
      </inscription>
      <graphics/>
      <toolspecific tool="WoPeD" version="1.0">
        <probability>1.0</probability>
        <displayProbabilityOn>false</displayProbabilityOn>
        <displayProbabilityPosition x="500.0" y="0.0"/>
      </toolspecific>
    </arc>
    <arc id="a21" source="p3" target="t5">
      <inscription>
        <text>1</text>
      </inscription>
      <graphics/>
      <toolspecific tool="WoPeD" version="1.0">
        <probability>1.0</probability>
        <displayProbabilityOn>false</displayProbabilityOn>
        <displayProbabilityPosition x="500.0" y="0.0"/>
      </toolspecific>
    </arc>
    <arc id="a13" source="t3" target="p5">
      <inscription>
        <text>1</text>
      </inscription>
      <graphics/>
      <toolspecific tool="WoPeD" version="1.0">
        <probability>1.0</probability>
        <displayProbabilityOn>false</displayProbabilityOn>
        <displayProbabilityPosition x="500.0" y="0.0"/>
      </toolspecific>
    </arc>
    <arc id="a10" source="p4" target="t3">
      <inscription>
        <text>1</text>
      </inscription>
      <graphics/>
      <toolspecific tool="WoPeD" version="1.0">
        <probability>1.0</probability>
        <displayProbabilityOn>false</displayProbabilityOn>
        <displayProbabilityPosition x="500.0" y="0.0"/>
      </toolspecific>
    </arc>
    <arc id="a24" source="t5" target="p2">
      <inscription>
        <text>1</text>
      </inscription>
      <graphics/>
      <toolspecific tool="WoPeD" version="1.0">
        <probability>1.0</probability>
        <displayProbabilityOn>false</displayProbabilityOn>
        <displayProbabilityPosition x="500.0" y="0.0"/>
      </toolspecific>
    </arc>
    <arc id="a1" source="p1" target="t1">
      <inscription>
        <text>1</text>
      </inscription>
      <graphics/>
      <toolspecific tool="WoPeD" version="1.0">
        <probability>1.0</probability>
        <displayProbabilityOn>false</displayProbabilityOn>
        <displayProbabilityPosition x="500.0" y="0.0"/>
      </toolspecific>
    </arc>
    <arc id="a2" source="t1" target="p2">
      <inscription>
        <text>1</text>
      </inscription>
      <graphics/>
      <toolspecific tool="WoPeD" version="1.0">
        <probability>1.0</probability>
        <displayProbabilityOn>false</displayProbabilityOn>
        <displayProbabilityPosition x="500.0" y="0.0"/>
      </toolspecific>
    </arc>
    <arc id="a5" source="p2" target="t2">
      <inscription>
        <text>1</text>
      </inscription>
      <graphics/>
      <toolspecific tool="WoPeD" version="1.0">
        <probability>1.0</probability>
        <displayProbabilityOn>false</displayProbabilityOn>
        <displayProbabilityPosition x="500.0" y="0.0"/>
      </toolspecific>
    </arc>
    <arc id="a6" source="t2" target="p3">
      <inscription>
        <text>1</text>
      </inscription>
      <graphics/>
      <toolspecific tool="WoPeD" version="1.0">
        <probability>1.0</probability>
        <displayProbabilityOn>false</displayProbabilityOn>
        <displayProbabilityPosition x="500.0" y="0.0"/>
      </toolspecific>
    </arc>
    <arc id="a18" source="t4" target="p6">
      <inscription>
        <text>1</text>
      </inscription>
      <graphics/>
      <toolspecific tool="WoPeD" version="1.0">
        <probability>1.0</probability>
        <displayProbabilityOn>false</displayProbabilityOn>
        <displayProbabilityPosition x="500.0" y="0.0"/>
      </toolspecific>
    </arc>
    <toolspecific tool="WoPeD" version="1.0">
      <bounds>
        <position x="11" y="33"/>
        <dimension x="755" y="490"/>
      </bounds>
      <treeWidth>2</treeWidth>
      <verticalLayout>false</verticalLayout>
      <resources/>
      <simulations/>
      <partnerLinks/>
      <variables/>
    </toolspecific>
  </net>
</pnml>
```

A matter of abstraction

XML	http://www.w3.org/XML/
SOAP	http://www.w3.org/TR/soap/
WSDL	http://www.w3.org/TR/wsdl20/
UDDI	http://www.uddi.org/

A technology should not become “THE model”
(though it can serve as a source of inspiration)

this way of thinking can have a bad impact
in the near future (in training technical personnel)




```
<operation  name = "CheckAvailability">
  < input message = "CheckInDate"/>
  < input message = "CheckOutDate"/>
  < input message = "NRooms"/>
  <output message = "Result"/>
</operation>
```

The source of the problem

BPEL is designed to work with WSDL documents of the services required by the process

A process can itself be exposed as a service which needs its own WSDL document

For us:
we can forget that WSDL documents are written in XML
we regard them as abstract interface descriptions

BPEL guidelines

Structured control vs free flow

BPEL4WS should provide
both hierarchical and graph-like control regimes,
and allow their usage to be blended
as seamlessly as possible.

About data handling

BPEL4WS provides limited data manipulation functions
that are sufficient for the simple manipulation of data
that is needed to define
process relevant data and control flow.

Correlation

BPEL4WS should support an
identification mechanism for process instances
that allows the definition of instance identifiers
at the application message level.

Instance identifiers should be partner defined
and may change over time.

Abstract vs executable

BPEL4WS should define a set of Web service orchestration concepts that are meant to be used in common by both the external (abstract) and internal (executable) views of a business process.

Such a business process defines the behavior of a single autonomous entity, typically operating in interaction with other similar peer entities.

It is recognized that each usage view will require a few specialized extensions, but these extensions are to be kept to a minimum and tested against requirements

Transactions

BPEL4WS should define a long-running transaction model that is based on practically proven techniques like compensation actions and scoping to support failure recovery for parts of long-running business processes.

WSDL preliminaries

Service

A service can be thought of as a container for a set of (logically related) operations that are made available via web-based protocols

Roughly: a remote object

PortType / Interface

The **`<portType>`** element,
renamed to **`<interface>`** in WSDL 2.0,
defines a web service,
the operations that can be performed,
and the messages that are used to perform the operation.

Roughly: the type of a remote object

i.e., a remote (abstract) class

Operation

Each operation can be thought of as
a method or function call in some programming language.

Four kinds of operations
(one-way, request-response, notification, solicit-response)
Three kinds of parameters/arguments
(input, output, fault)
(not all combinations allowed)

Roughly: a remote (abstract) method

Port / Endpoint

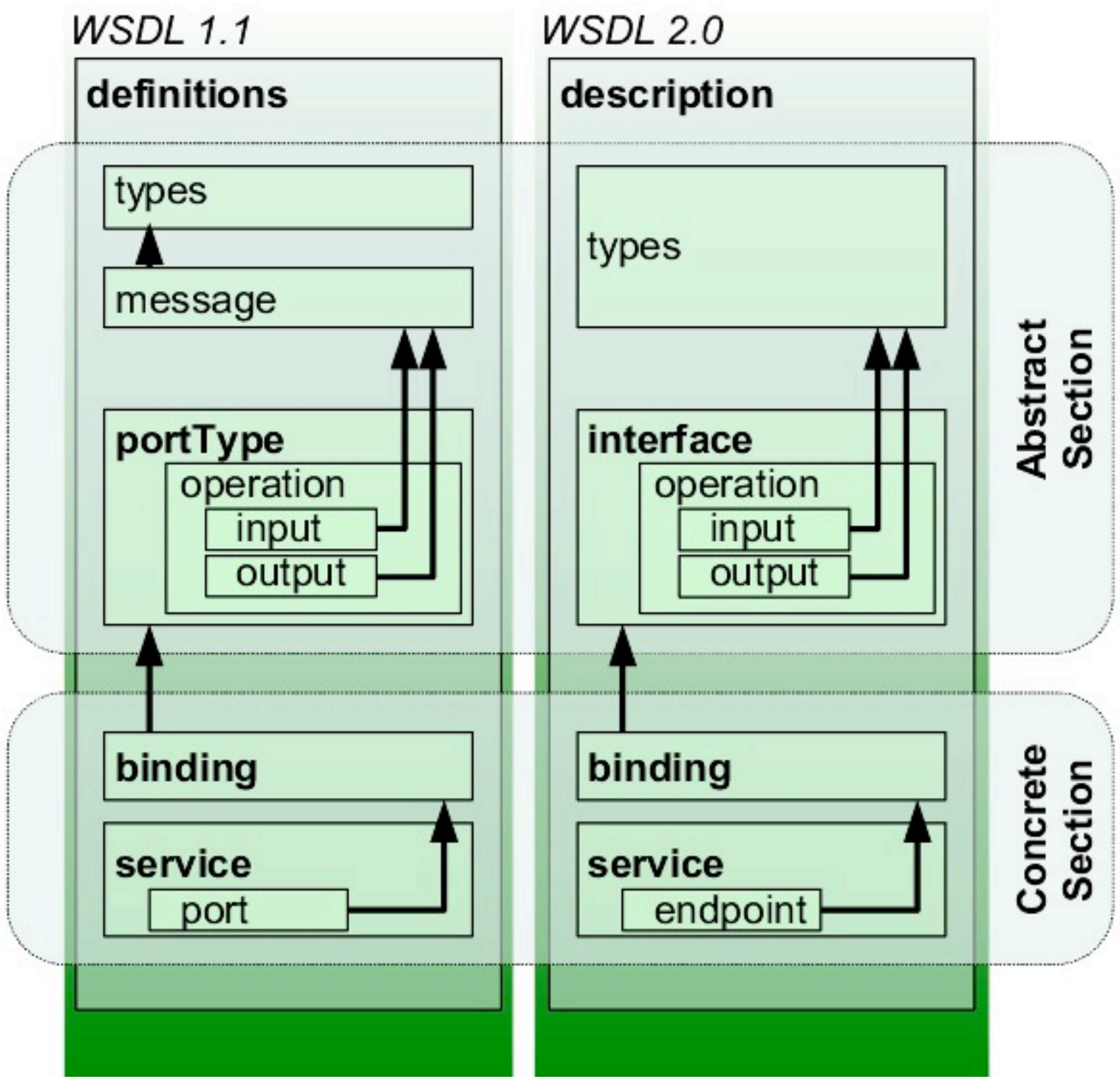
The **`<port>`** element,
renamed to **`<endpoint>`** in WSDL 2.0,
declares the address of a web service.

It typically involves a name, a binding and a URL

Binding

The binding specifies the interface as well as the SOAP binding style (message format) and SOAP transport protocol.

WSDL (from wikipedia)



```xml
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="PurchaseExample"
    targetNamespace="http://www.fluidimagination.com/sams/PurchaseExample.wsdl"
    xmlns:tns="http://www.fluidimagination.com/sams/PurchaseExample.wsdl"
    xmlns:soap="http://www.schemas.xmlsoap.org/wsdl/soap/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">

    <wsdl:types>
        <xsd:schema
                  targetNamespace="http://www.fluidimagination.com/sams/productType.wsdl"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
            <xsd:complexType name="scannerType">
                <xsd:all>
                    <xsd:element name="upc" type="upcType"/>
                    <xsd:element name="isbn" type="isbnType"/>
                </xsd:all>
            </xsd:complexType>
            <xsd:simpleType name="upcType">
                <xsd:restriction base="xsd:string">
                    <xsd:pattern value="[0-9]{12}"/>
                </xsd:restriction>
            </xsd:simpleType>
            <xsd:simpleType name="isbnType">
                <xsd:restriction base="xsd:string">
                    <xsd:pattern value="([0-9]- ){10}"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:schema>
    </wsdl:types>
    <!-- Adding a message that has two addresses -->
    <wsdl:message name="purchaseMessage">
        <wsdl:part name="productCode" element="tns:scannerType"/>
    </wsdl:message>
    <!--create a port type with one operation -->
    <wsdl:portType name="purchaseType">
        <wsdl:operation name="purchaseOperation">
            <wsdl:input name="tns:purchaseMessage"/>
        </wsdl:operation>
    </wsdl:portType>
    <!--Bind the message to SOAP using HTTP -->
    <wsdl:binding name="purchaseBinding" type="tns:purchaseType">
        <soap:binding style="document"
              transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="tns:purchaseOperation">
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
        </wsdl:operation>
    </wsdl:binding>
    <!--Bind the message to SOAP over  SMTP -->
    <wsdl:binding name="purchaseBindingSMTP" type="tns:purchaseType">
        <soap:binding style="document"
              transport="http://schemas.xmlsoap.org/soap/smtp"/>
        <wsdl:operation name="tns:purchaseOperation">
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
        </wsdl:operation>
    </wsdl:binding>
</wsdl:definitions>
```

BPEL ingredients

(material partly “stolen” from Antonio Brogi’s slides on Software Services, thanks!)

BPEL ingredients

Data flow
(scoped variables)

Partner links and Message correlation

Message flow
(one-way, request-response, notify, solicit-response)

Control flow
(structured activities and synchronization links)

Handling events, faults, compensations

Variable

Variables can be defined (within a local scope)

The activity **`<assign>`** can be used to copy data (messages, part of messages, service references) between variables

```
<assign>
     <copy>
          <from variable="PO" part="customerInfo"/>
          <to variable="shippingRequest" part="customerInfo"/>
     </copy>
</assign>
```

Partner Link

A partner is a service that the process invokes, or a client that invokes the process

A BPEL process interacts with a partner using a **`<partnerLink>`** a (typed) connector that the process offers to/requires from its partner (to be declared in the BPEL document)

```
<partnerLinks>
    <partnerLink name="shipping"
                 partnerLinkType="lns:shippingLT"
                 myRole="shippingRequester"
                 partnerRole="shippingService"/>
    ...
</partnerLinks>
```

Stateless services, stateful processes

When a message for (WS-BPEL) service arrives,
it must be delivered either to a new
or to an existing instance of the process

Stateful business processes are instantiated to act
according to interaction history

Messages should not only be delivered to the correct port,
but also to the correct instance of the business process
that provides that port

Message correlation

Message correlation is the way to tie together messages coming from different communications

A correlation set is a set of properties such that all messages having the same values of all properties are part of the same interaction

The partner that first fixes the values of the properties in the correlation set is the **initiator** of the exchange, the other partners are called the **followers**

Message flow

Basic activities are available
to send and receive messages to partners

Activity **`<invoke>`**: asynchronous (one-way) or synchronous (request-response)

Activity **`<receive>`**: a request from a partner to execute one of the (WSDL) operations implemented by the process

Activity **`<reply>`**: to return the result of a **`<receive>`**d synchronous request-response operation

Invoke

Needed information: the **`<partnerLink>`**,
the WSDL **`<portType>`** of the service to be invoked, and
the name and parameters of the **`<operation>`**

```
<invoke partnerLink="shipping"
        portType="lns:shippingPT"
        operation="requestShipping"
        inputVariable="shippingRequest"
        outputVariable="shippingInfo">
        <source linkName="ship-to-invoice"/>
</invoke>
```

Receive

Needed information: the **`<partnerLink>`**, the WSDL **`<portType>`** of the exposed service, and a **`<variable>`** where to copy the parameters of the **`<operation>`**

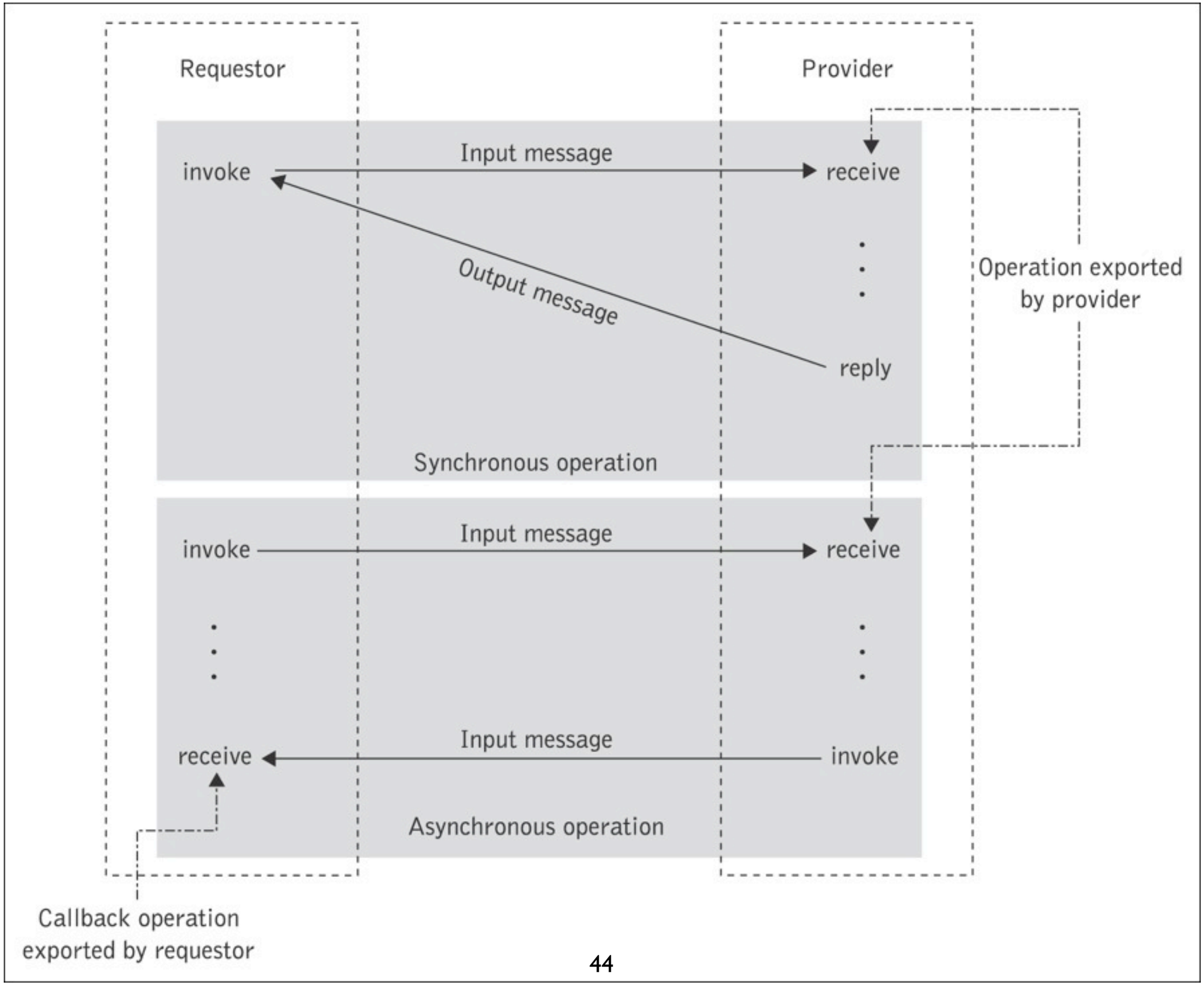
```
<receive partnerLink="purchasing"
         portType="lns:purchaseOrderPT"
         operation="sendPurchaseOrder"
         variable="PO">
</receive>
```

Reply

A process can **`<reply>`** to a message it **`<receive>`**d

```
<reply partnerLink="purchasing"
       portType="lns:purchaseOrderPT"
       operation="sendPurchaseOrder"
       variable="Invoice" />
```

Asynchronous operations do not use **`<reply>`**
If a reply must sent,
`<invoke>` is used to call back a client operation


Requestor
Provider
invoke
Input message
receive
Output message
reply
Operation exported
by provider
Synchronous operation
invoke
Input message
receive
receive
Input message
invoke
Asynchronous operation
Callback operation
exported by requestor

Structured activities

`<sequence>` for specifying sequential compositions

only one branch is selected

`<switch>` for (local) internal choices
(ordered list of conditional **`<case>`** branches,
possibly ended by an **`<otherwise>`** branch)

`<pick>` for (global) external choices
(set of event handlers of the form event → activity,
`<onMessage>` arrival of a message or **`<onAlarm>`** timer)

`<flow>` for parallel composition

`<while>` for iterations (guards are XPath expressions)

Link

A **`<link>`** expresses synchronisation dependencies among activities in a process

Each **`<link>`** has a name,
one source activity, one target activity, and
it may be associated with a transition condition
(a predicate to be evaluated when the source activity ends)

Join condition

Any activity that is the target of one or more links
may have an explicit **`<joinCondition>`**,
(a predicate on the status values of the incoming links,
to be evaluated once all such values have been determined)

otherwise the implicit join condition is the OR

If the **`<joinCondition>`** evaluates to:

TRUE the activity can be executed,

FALSE a **`<joinFailure>`** fault may be thrown
(depending on the **`<suppressJoinFailure>`** flag

Scope

A scope provides fault and compensation handling capabilities to the activities nested within it

A **`<scope>`** activity consists of:
a primary activity,

a set of (optional) fault handlers,

a single (optional) compensation handlers,

a set of (optional) event handlers
(executed concurrently with the process,
they enable a scope to react to messages and alarm events)

Formal Semantics and Analysis of Control Flow in WS-BPEL

(Revised Version)

Chun Ouyang[1], Eric Verbeek[2], Wil M.P. van der Aalst[2,1], Stephen Breutel[1], Marlon Dumas[1], and Arthur H.M. ter Hofstede[1]

[1] Faculty of Information Technology, Queensland University of Technology, GPO Box 2434, Brisbane QLD 4001, Australia
{c.ouyang,sw.breutel,m.dumas,a.terhofstede}@qut.edu.au

[2] Department of Technology Management, Eindhoven University of Technology, GPO Box 513, NL-5600 MB, The Netherlands
{h.m.w.verbeek,w.m.p.v.d.aalst}@tm.tue.nl

Formal semantics of control flow in BPEL

Motivation

BPEL specification:
rigourous XML syntax
English prose semantics (of apparent clarity)

Consequences:
inconsistencies, ambiguities, incompleteness

try to google for “WS BPEL issues list”, e.g.

Issue 32 Link Semantics in Event Handlers (resolved)
Issue 39 Inconsistent syntax for query attribute values in spec examples (resolved)

...

Issue 42 Need for Formalism (resolved) YES

Approaches

Promela (SPIN)

Process algebras

Abstract State Machines

Automata

Weakest preconditions / strongest postconditions

Axiomatic semantics

Petri nets

Goal

Unveil ambiguities in BPEL specification
(reported to BPEL standardization committee)

Complete formalization of all control-flow constructs

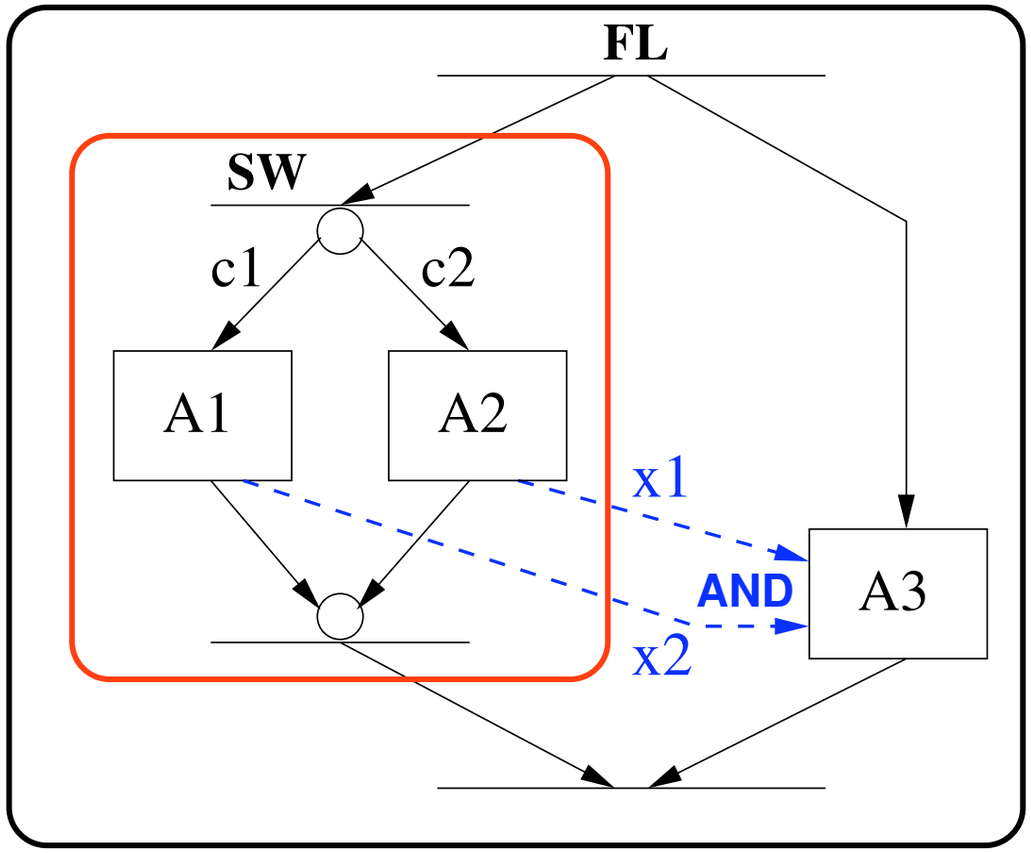
Checking for unreachable activities

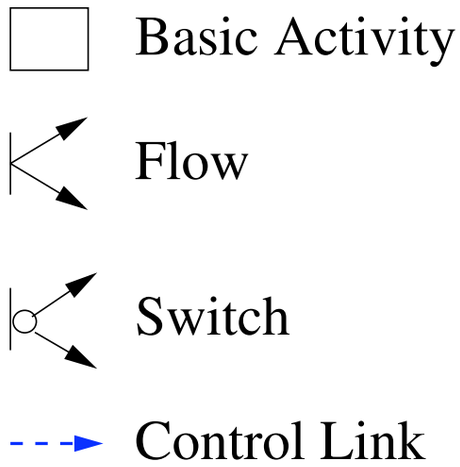
Checking for potential conflicting message receipt actions

Determining which messages can be eventually consumed

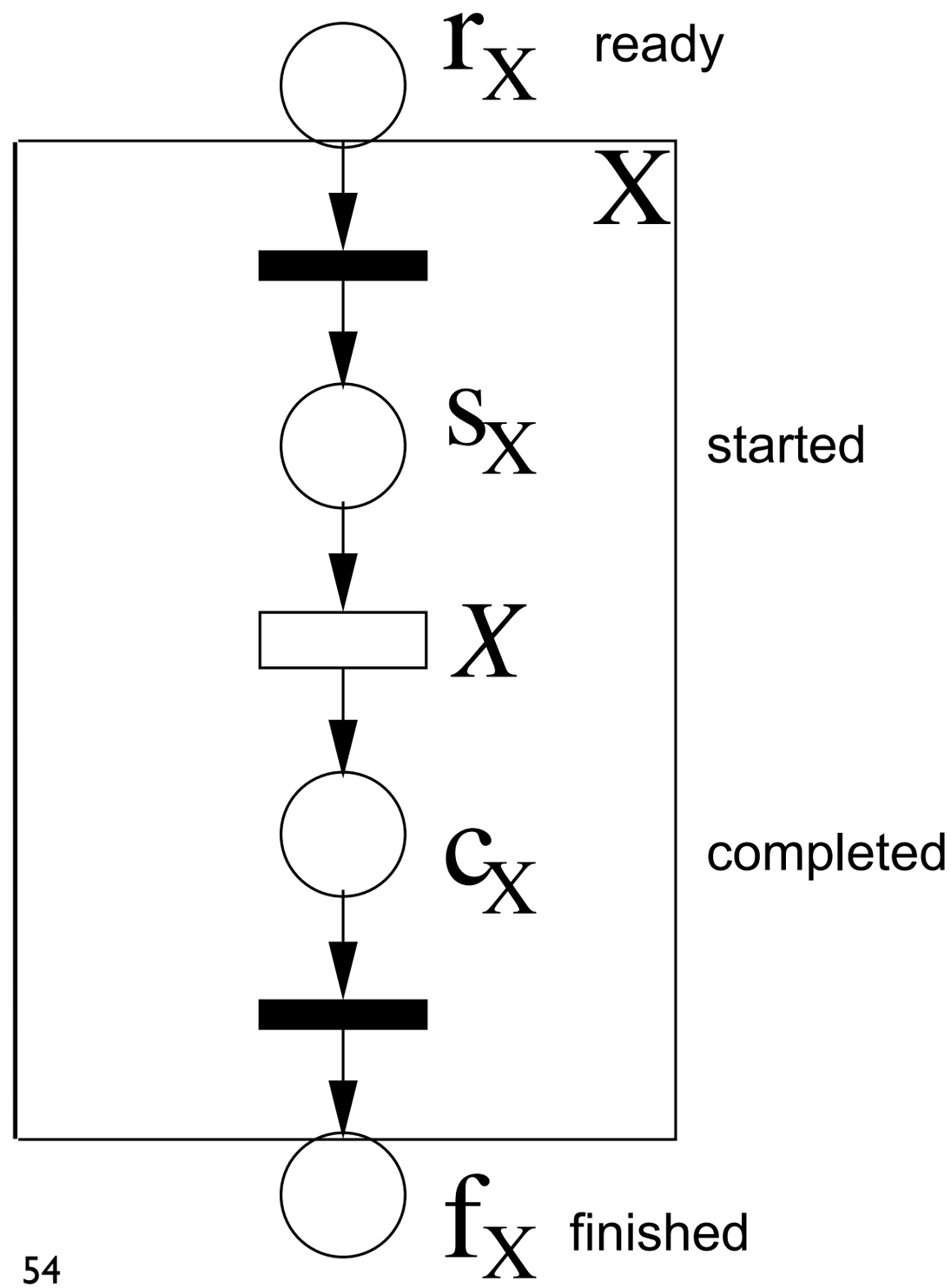
Example: BPEL with unreachable activity

```
<process name="unreachableTask"
    targetNamespace="http://samples.otn.com"
    suppressJoinFailure="yes"
    xmlns:tns="http://samples.otn.com"
    xmlns:services="http://services.otn.com"
    xmlns="http://schemas.xmlsoap.org/ws/2003/03/business−process/">
  <flow name="FL" suppressJoinFailure="yes">
  <links>
    <link name="x1"/>
    <link name="x2"/>
  </links>
  <switch name="SW">
    <case>
      <invoke name="A1">
        <sources>  <source linkName="x1"/>   </sources>
      </invoke>
    </case>
    <otherwise>
      <invoke name="A2">
        <sources>  <source linkName="x2"/>   </sources>
      </invoke>
    </otherwise>
  </switch>
  <invoke name = "A3">
    <targets>
    <joinCondition>
      bpws:getLinkStatus(‘x1’) and bpws:getLinkStatus(‘x2’)
    </joinCondition>
    <target linkName="x1"/>
    <target linkName="x2"/>
    </targets>
  </invoke>
  </flow>
</process>
```





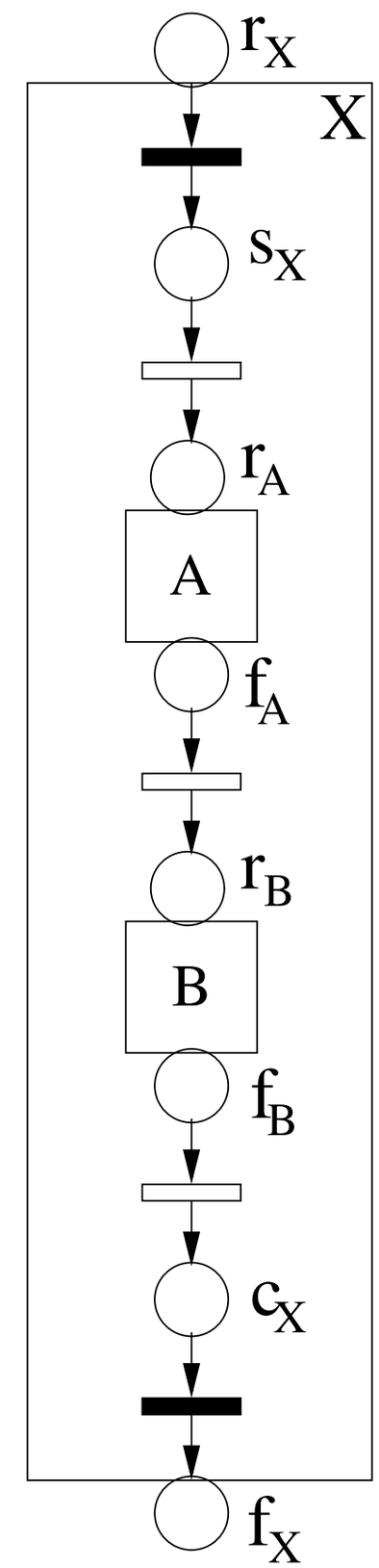
Basic activity X



Sequence A;B

We show the binary version, but it can be generalized to an arbitrary number of activities

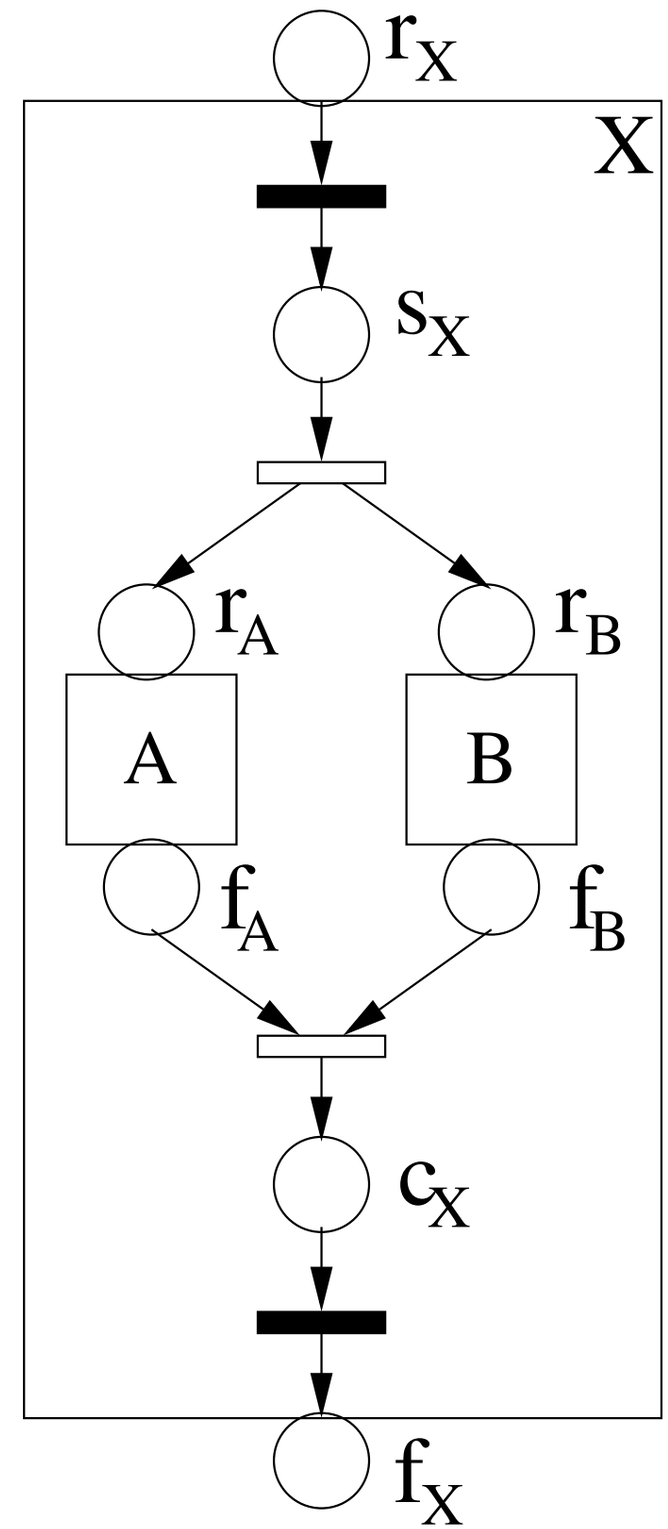
```
<sequence
  name="X">
  activity A
  activity B
</sequence>
```



Flow A|B

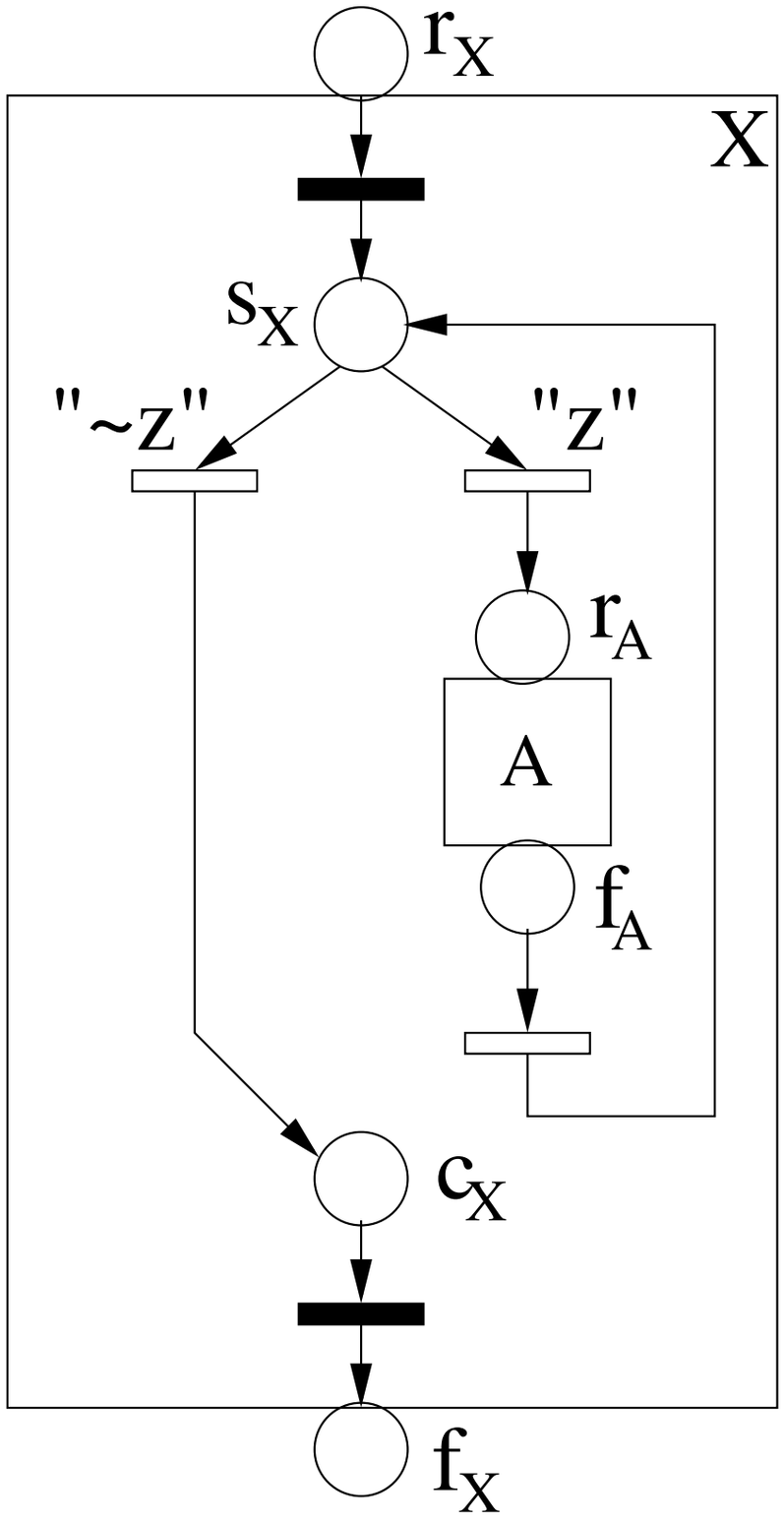
We show the binary version, but it can be generalized to an arbitrary number of activities

```
<flow
  name="X">
   activity A
   activity B
</flow>
```



While z do A

```
<while name="X">
    <condition>
        z
    </condition>
    activity A
</while>
```



Switch $(z_1)A,(z_2)B$

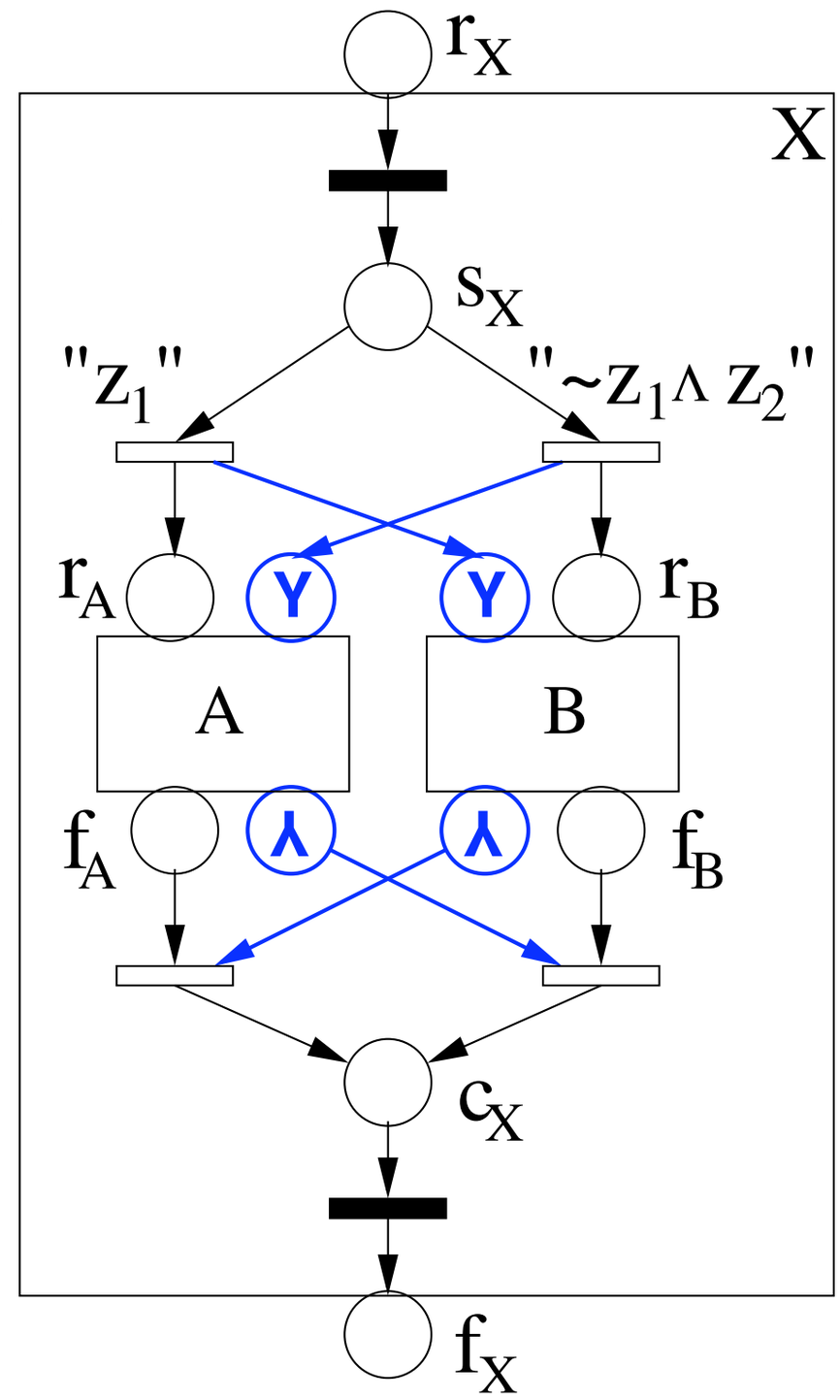
We show the binary version, but it can be generalized to an arbitrary number of activities

In blue: alternative flow to skip activities (also needed for links)

just decorations

Y ⅄

```
<switch name="X">
  <case>
    <condition>
      z1
    </condition>
    activity A
  </case>
  <case>
    <condition>
      z2
    </condition>
    activity B
  </case>
</switch>
```



Pick (e_1)A,(e_2)B

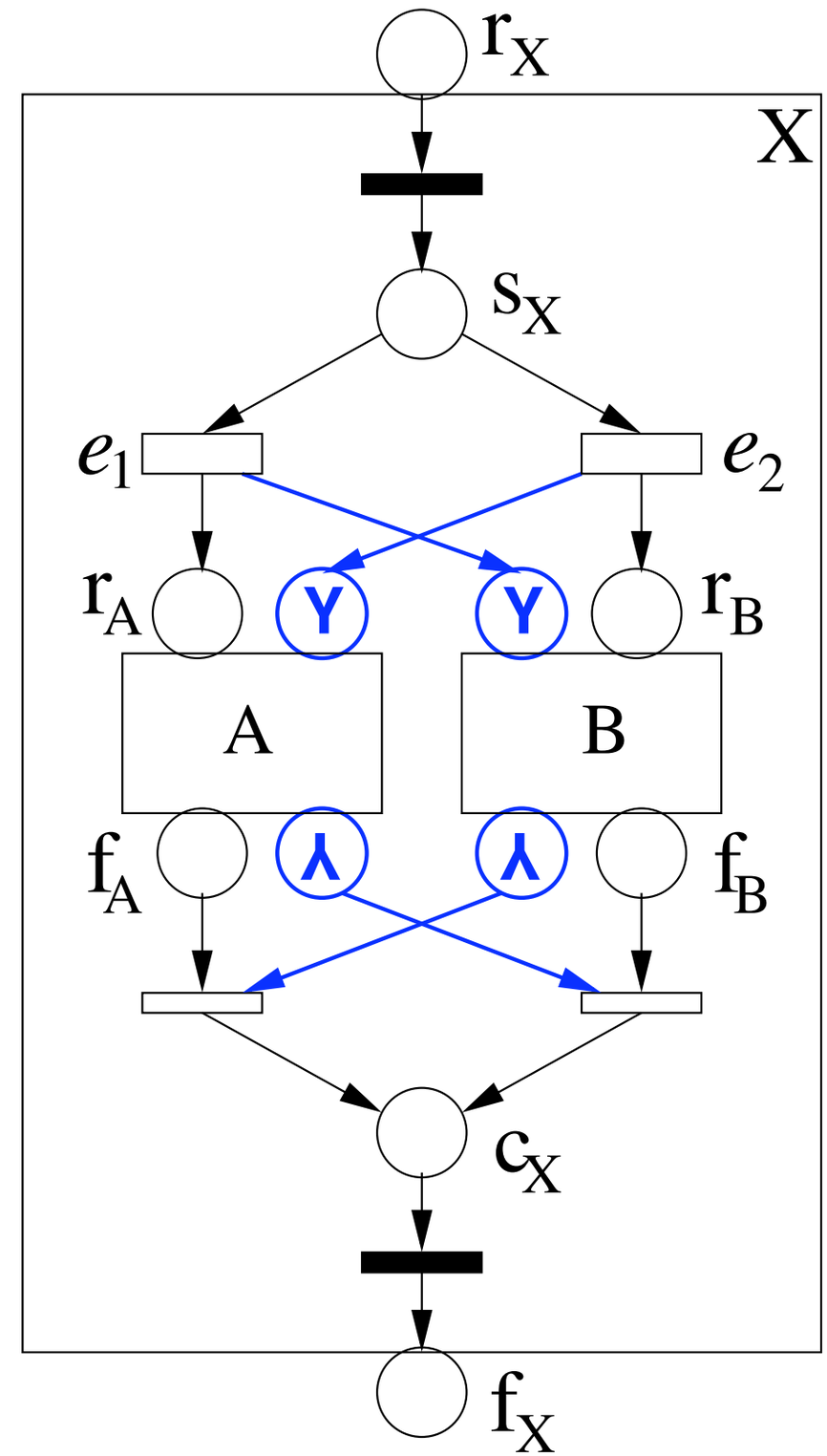
We show the binary version, but it can be generalized to an arbitrary number of activities

In blue: alternative flow to skip activities (also needed for links)

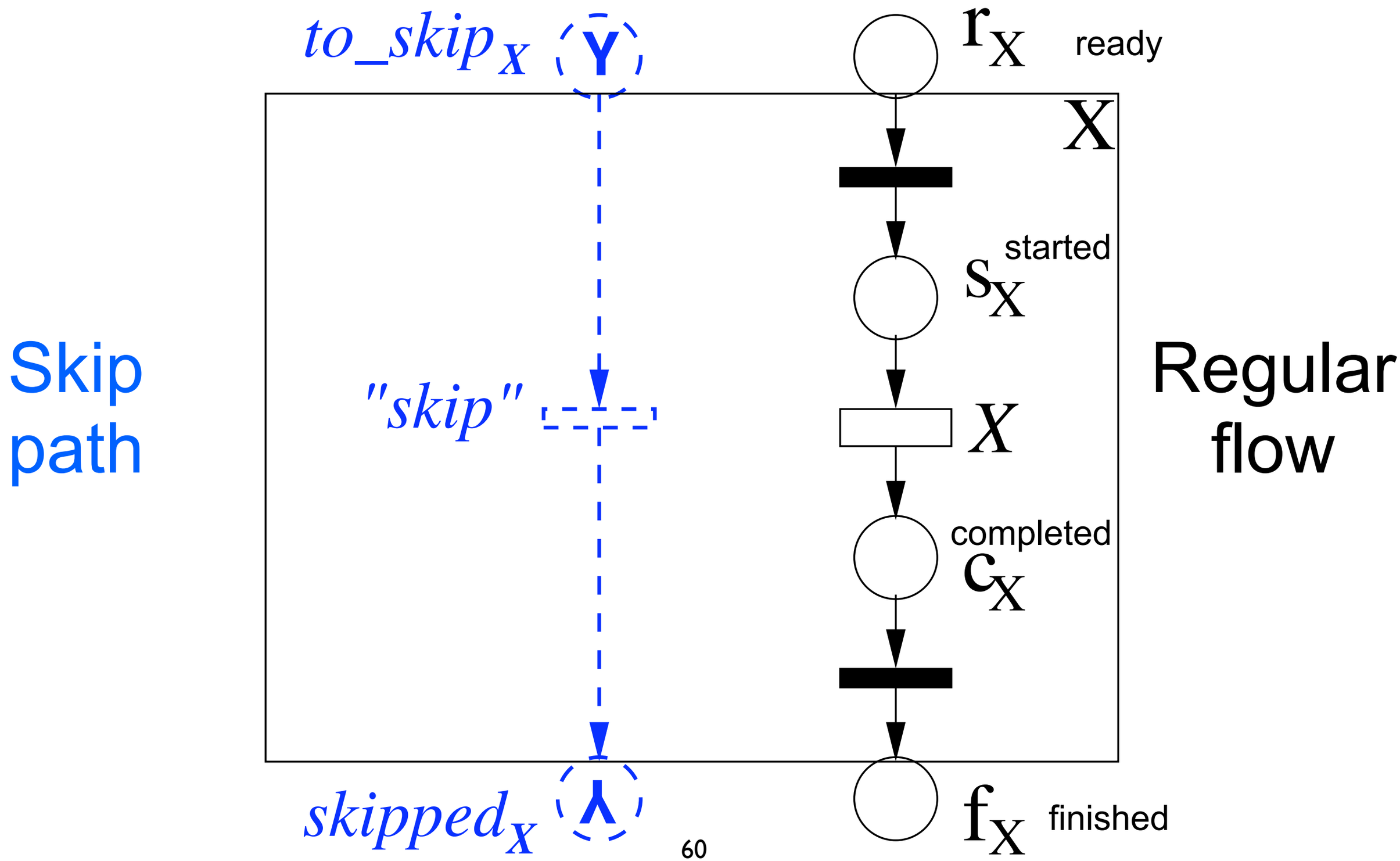
just decorations

Y λ

```
<pick name="X">
  <onMessage e1>
    activity A
  </onMessage>
  <onAlarm e2>
    activity B
  </onAlarm>
</pick>
```

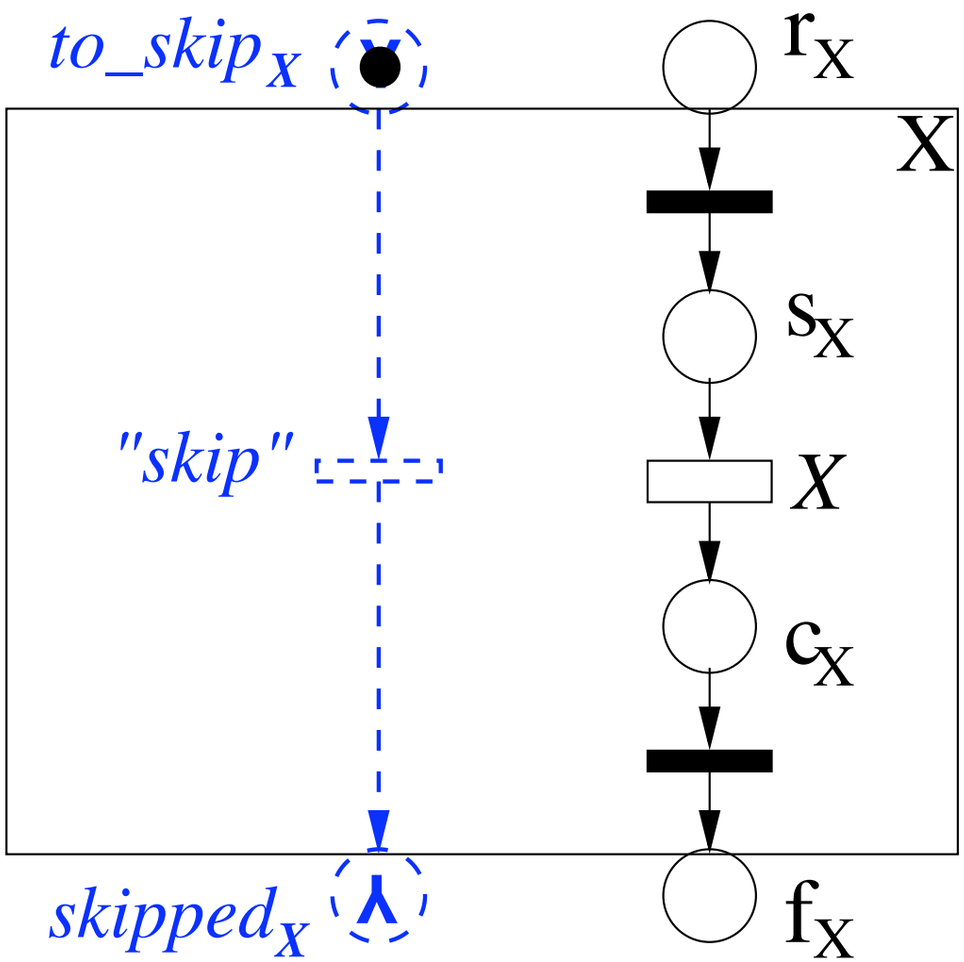


Basic activity + skip

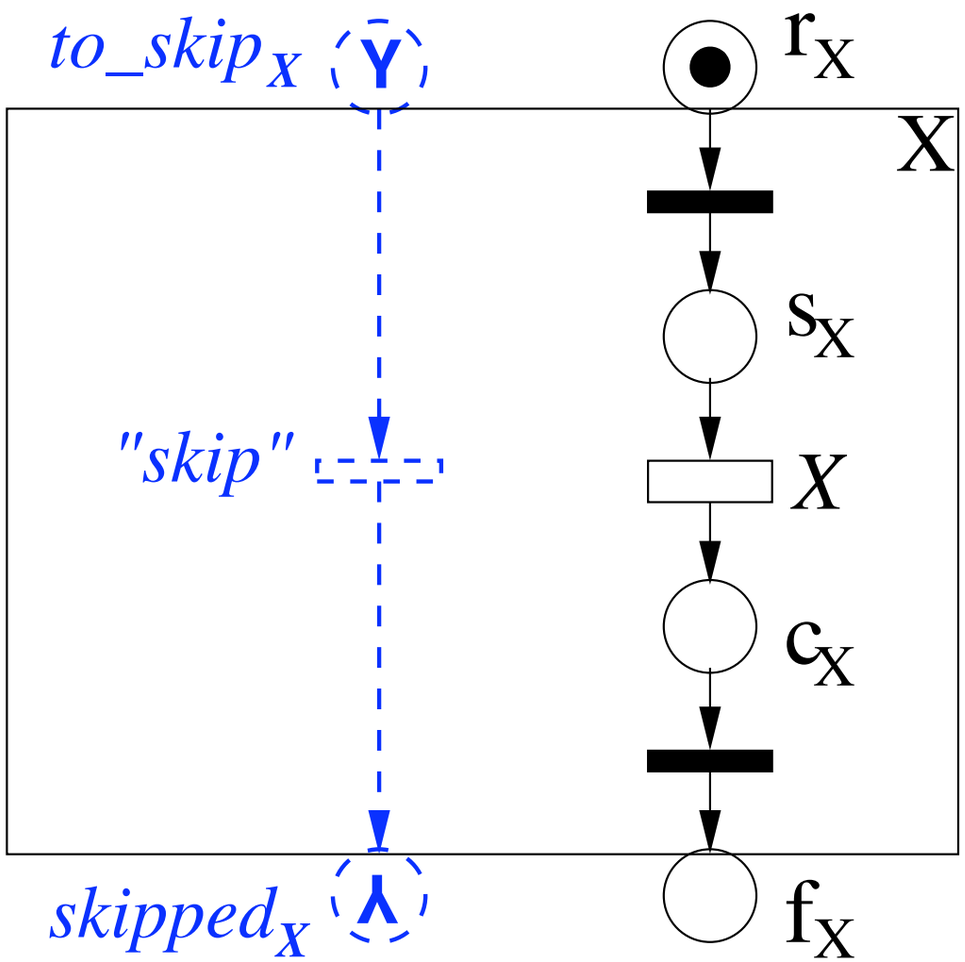


Basic activity + skip

The token arrives
either here...



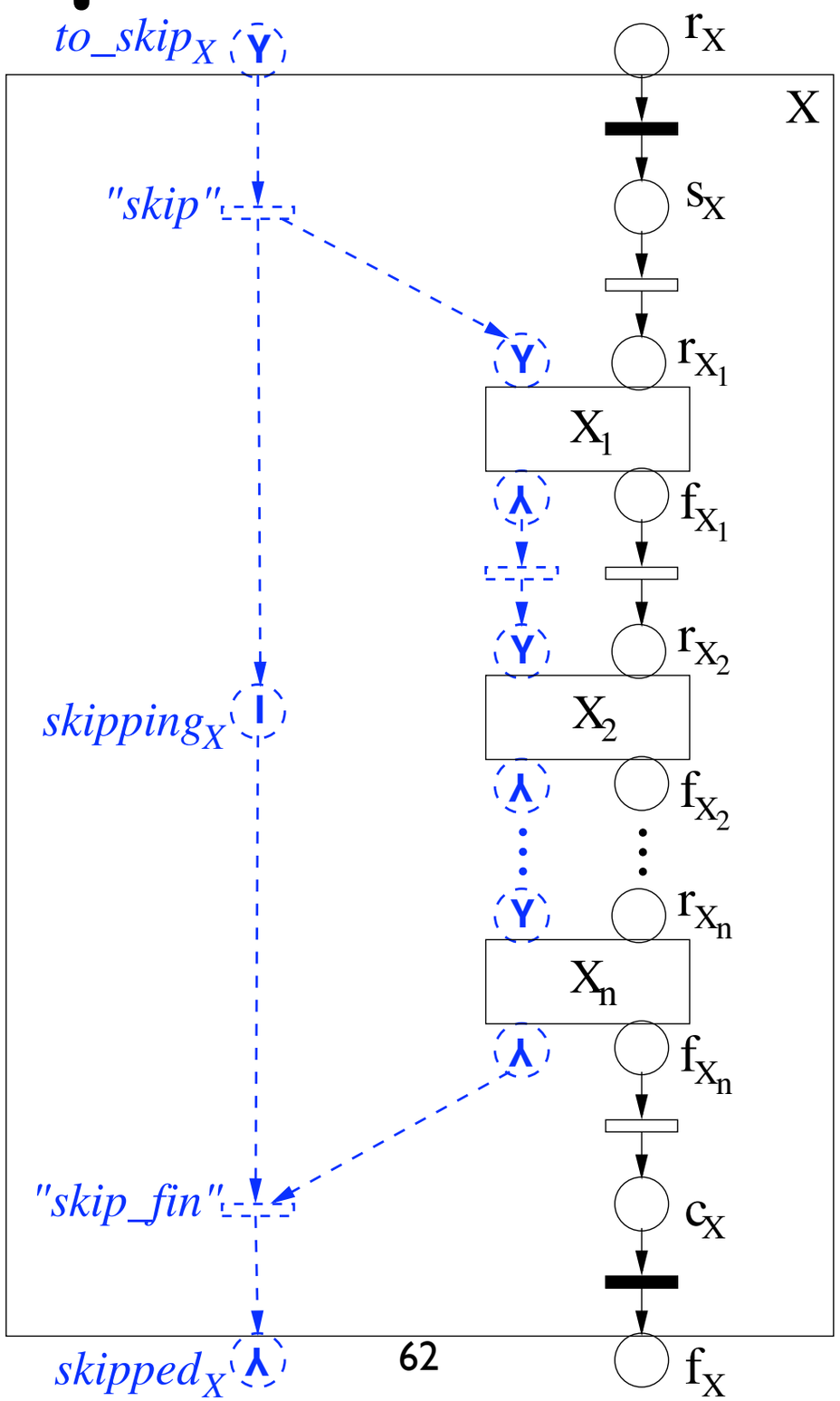
... or here



(but not both)

Sequence + skip

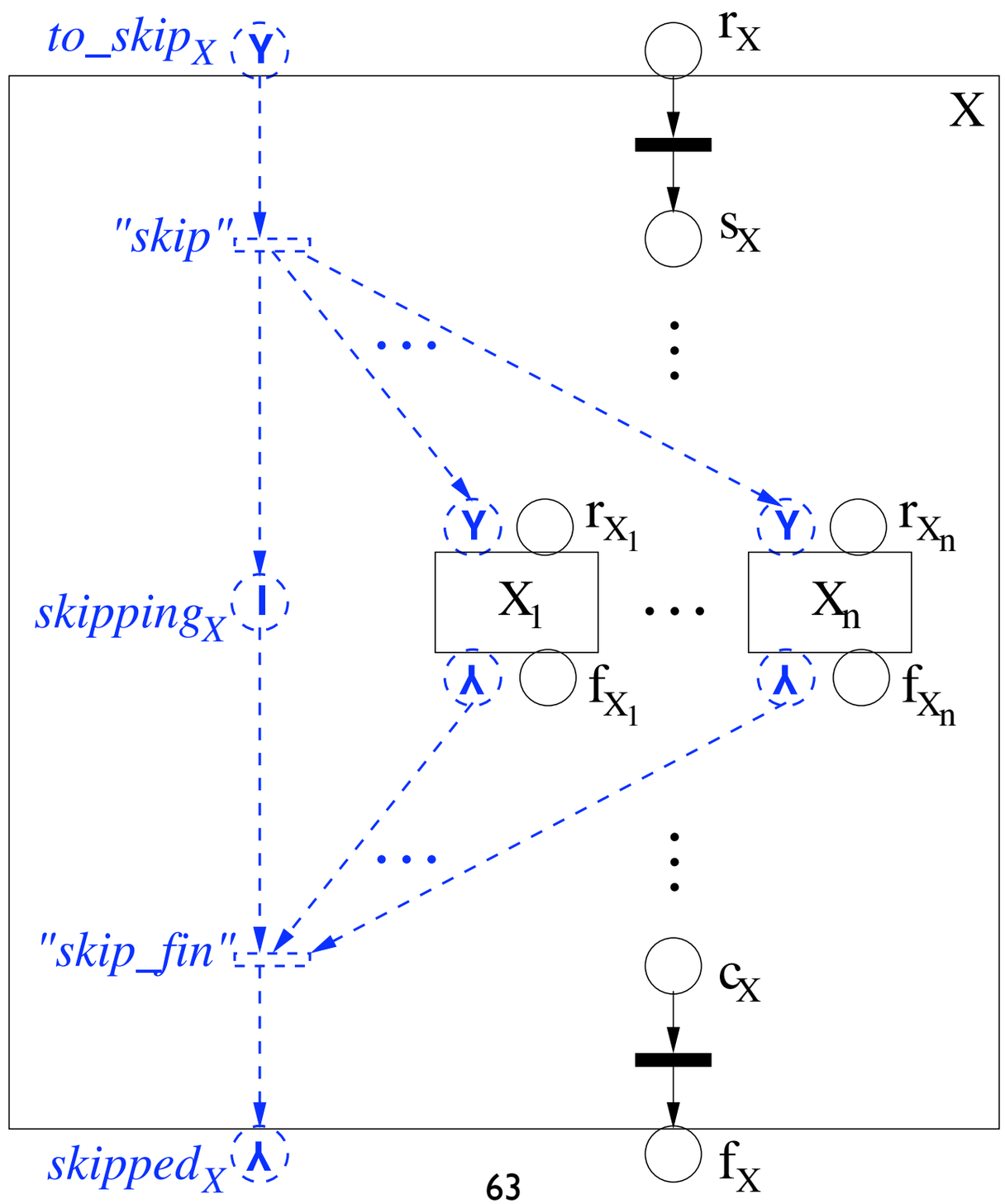
Skip path



Regular flow

Non-sequence + skip

Skip path



Regular flow

What about control links?

Control links are a non-structural element that introduces control dependencies

An activity can be the source of many links (it must evaluate the corresponding “transition condition”)

An activity can be the target of many links (it must receive their boolean evaluation and apply the join condition)

Join condition failure

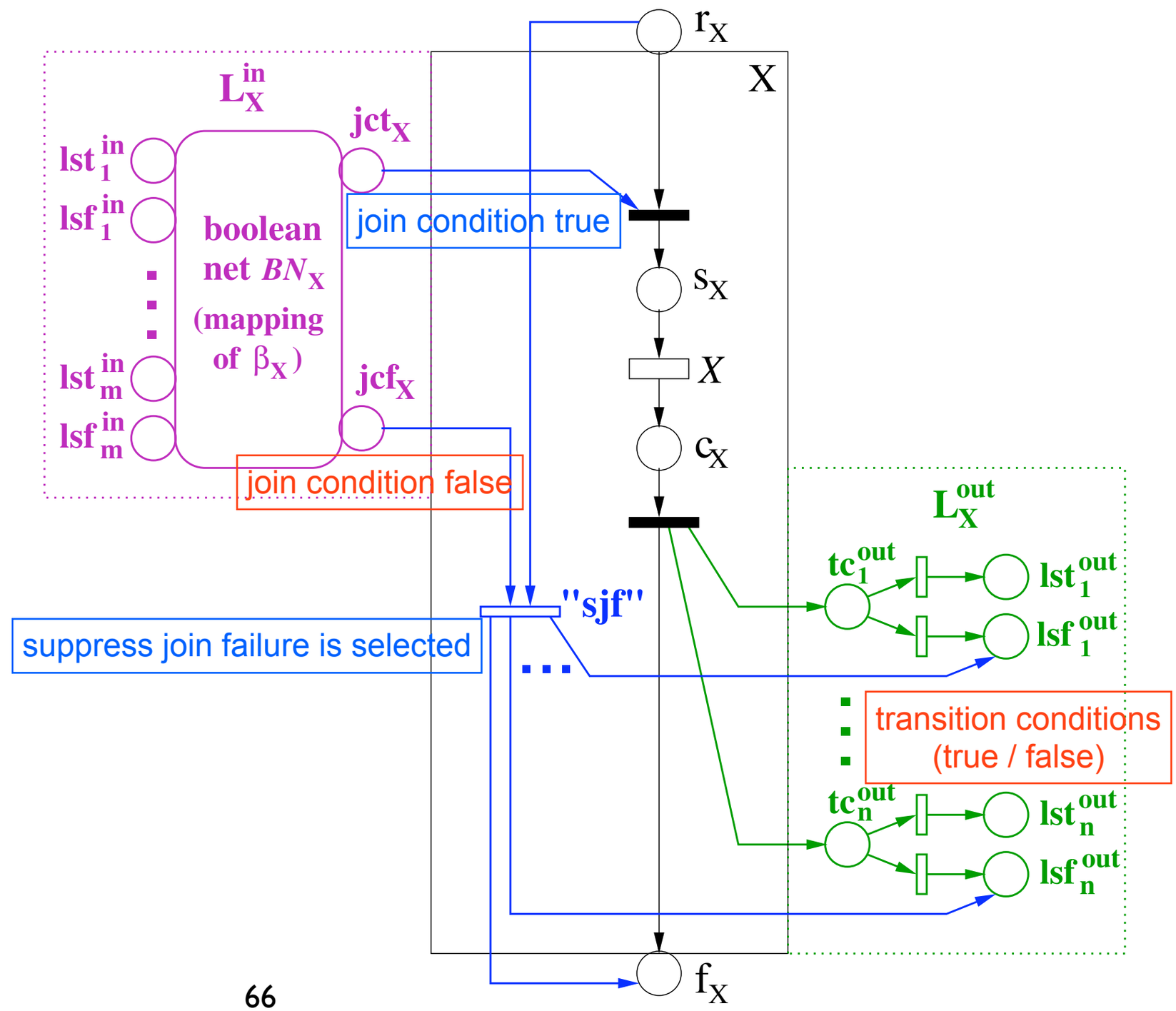
If the attribute **`suppressJoinFailure`** is set to **`no`**,
a join failure needs to be thrown,
which triggers a standard fault handling procedure

If the attribute **`suppressJoinFailure`** is set to **`yes`**,
the activity will not be performed,
will end up in the “finished” state,
(the processing of any following activity will not be affected)
and the status of all outgoing links will be set to **`false`**.
This is known as **dead path elimination**
(the false link status is propagated transitively along the paths formed by control links, until a join condition is reached that evaluates to true)

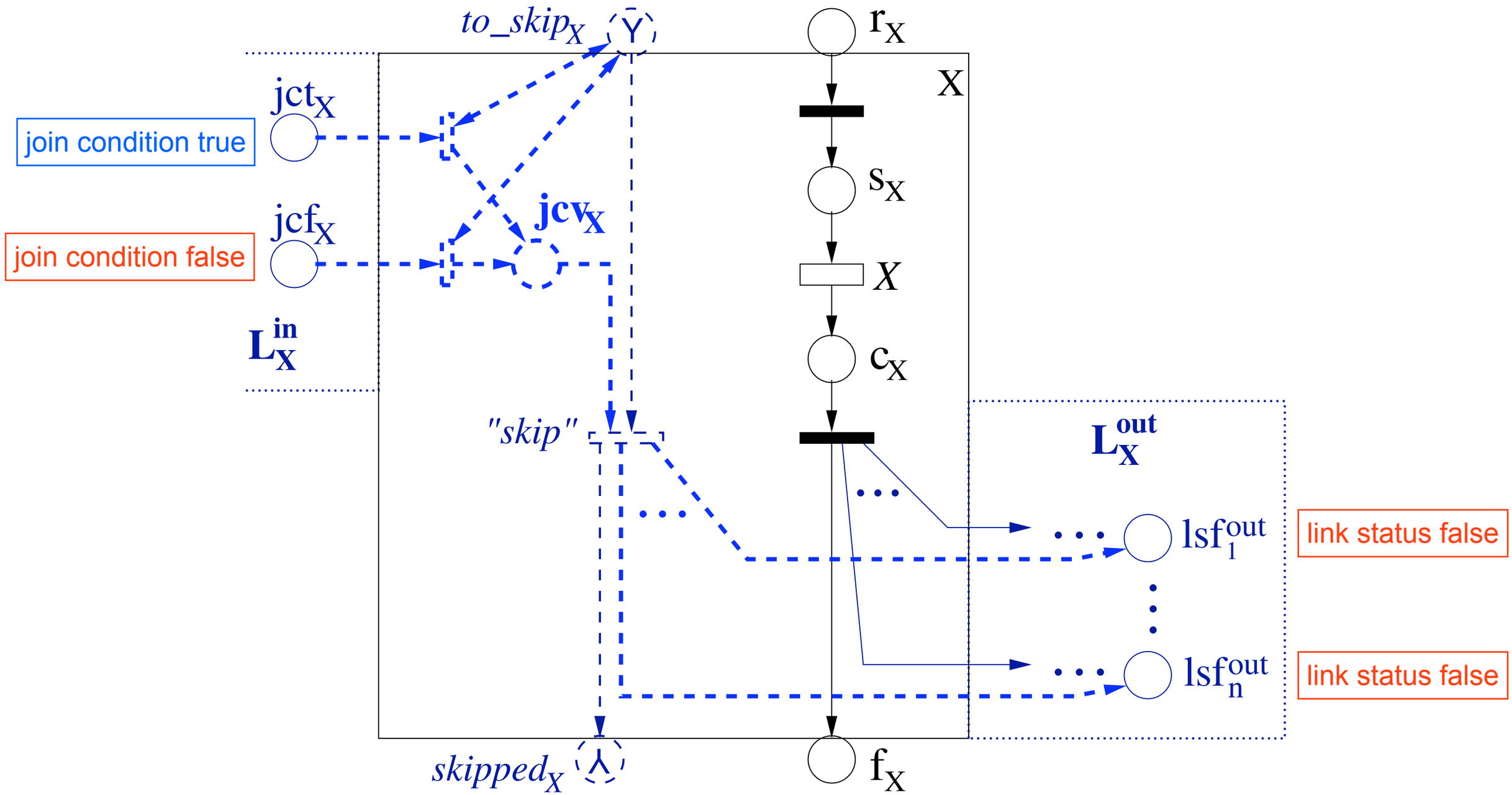
Basic activity with control links

```
<activityX suppressJoinFailure="yes">
  <sources>
    <source linkname="X_1^out">
      <transitionCondition>
          tc_1^out
      </transitionCondition>
    </source>
      ⋮
    <source linkname="X_n^out">
      <transitionCondition>
          tc_n^out
      </transitionCondition>
    </source>
  </sources>
  <targets>
    <joinCondition>
      β_X (ls_1^in , ... , ls_m^in )
    </joinCondition>
    <target linkname="X_1^in">
      ⋮
    <target linkname="X_m^in">
  </targets>
</activityX>
```

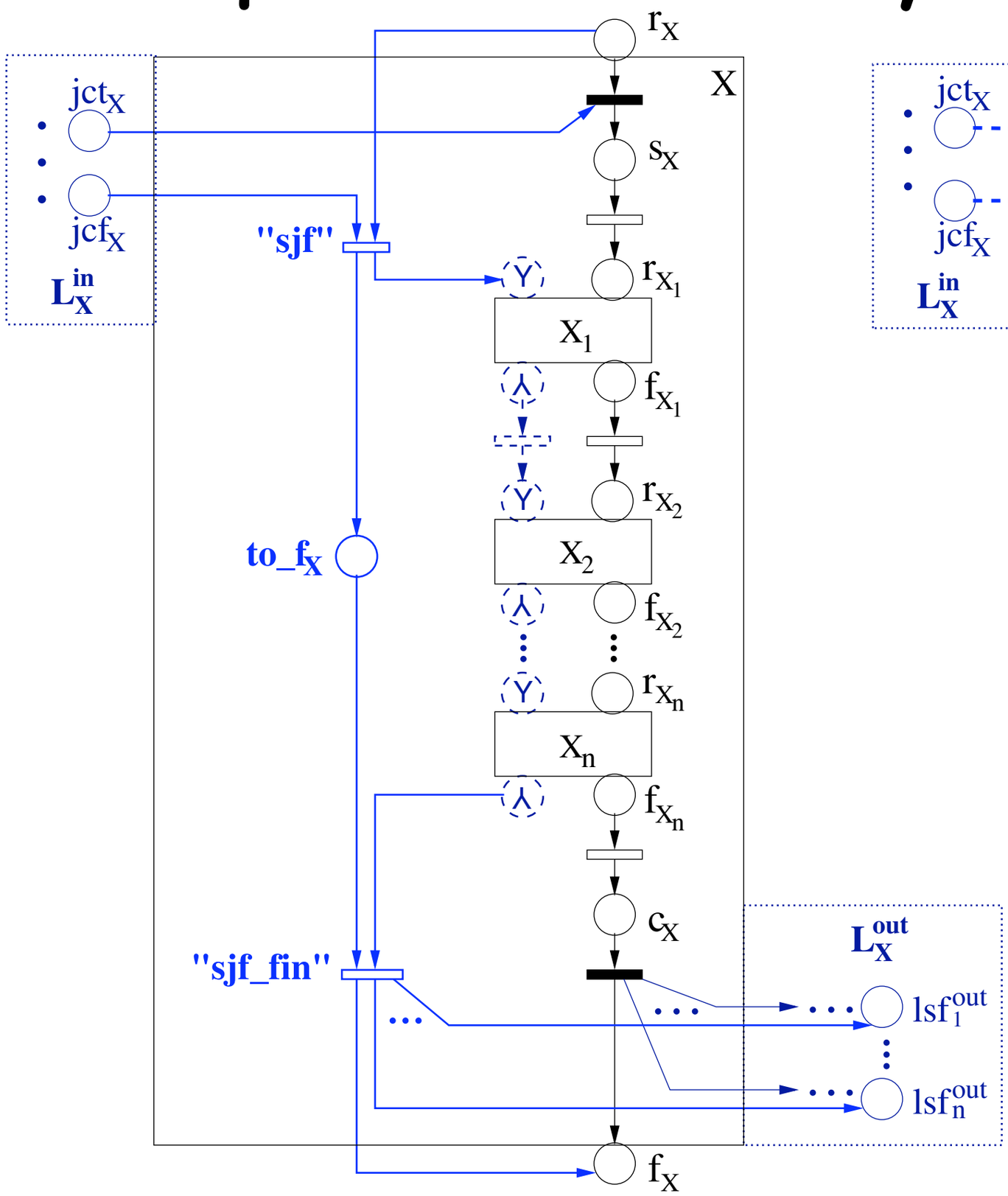
[note] ls_j^{in} *is the status of control link* X_j^{in}, *where j=1, 2, ... , m.*



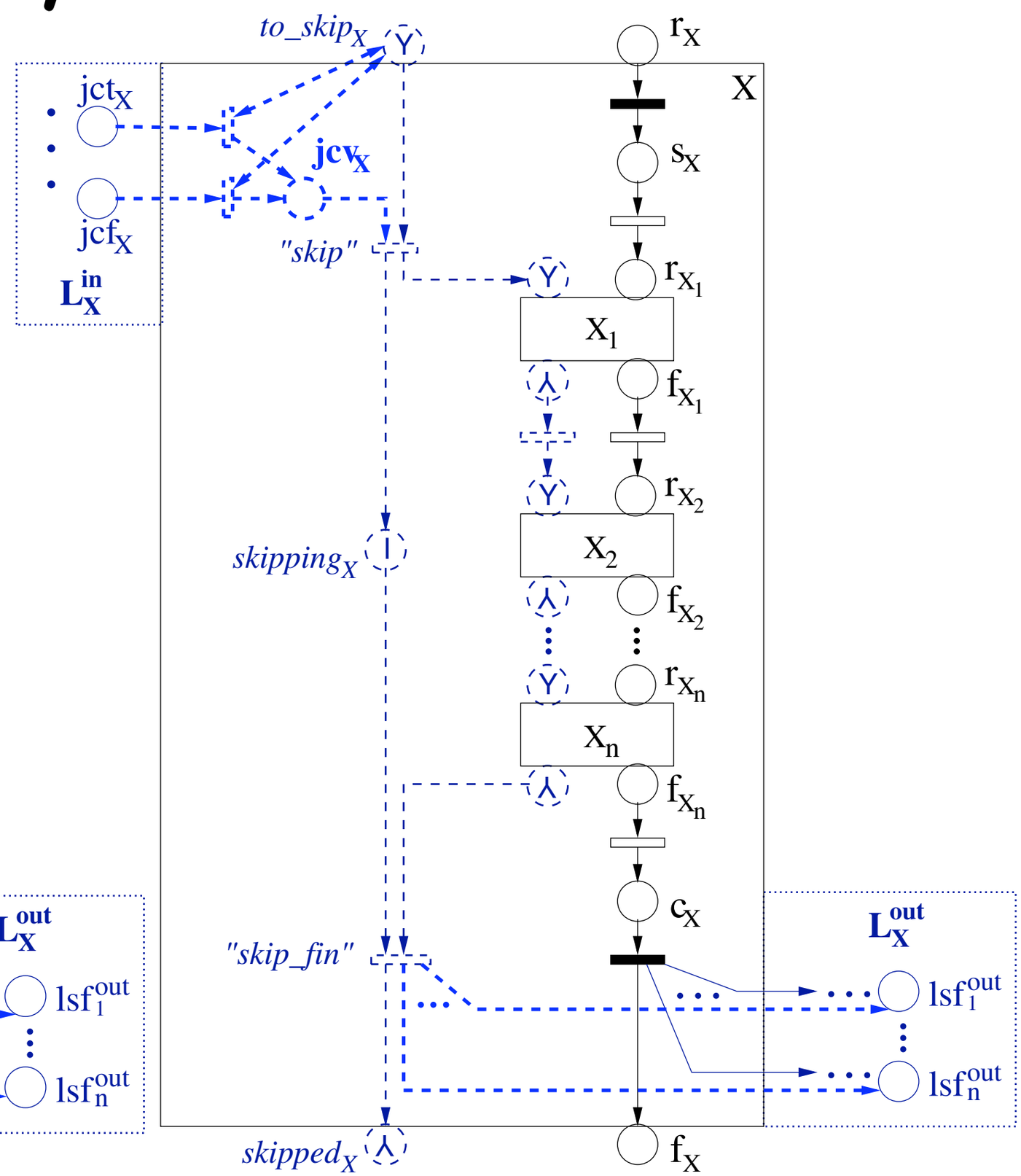
Skipping a basic activity with control links



Sequence activity with control links

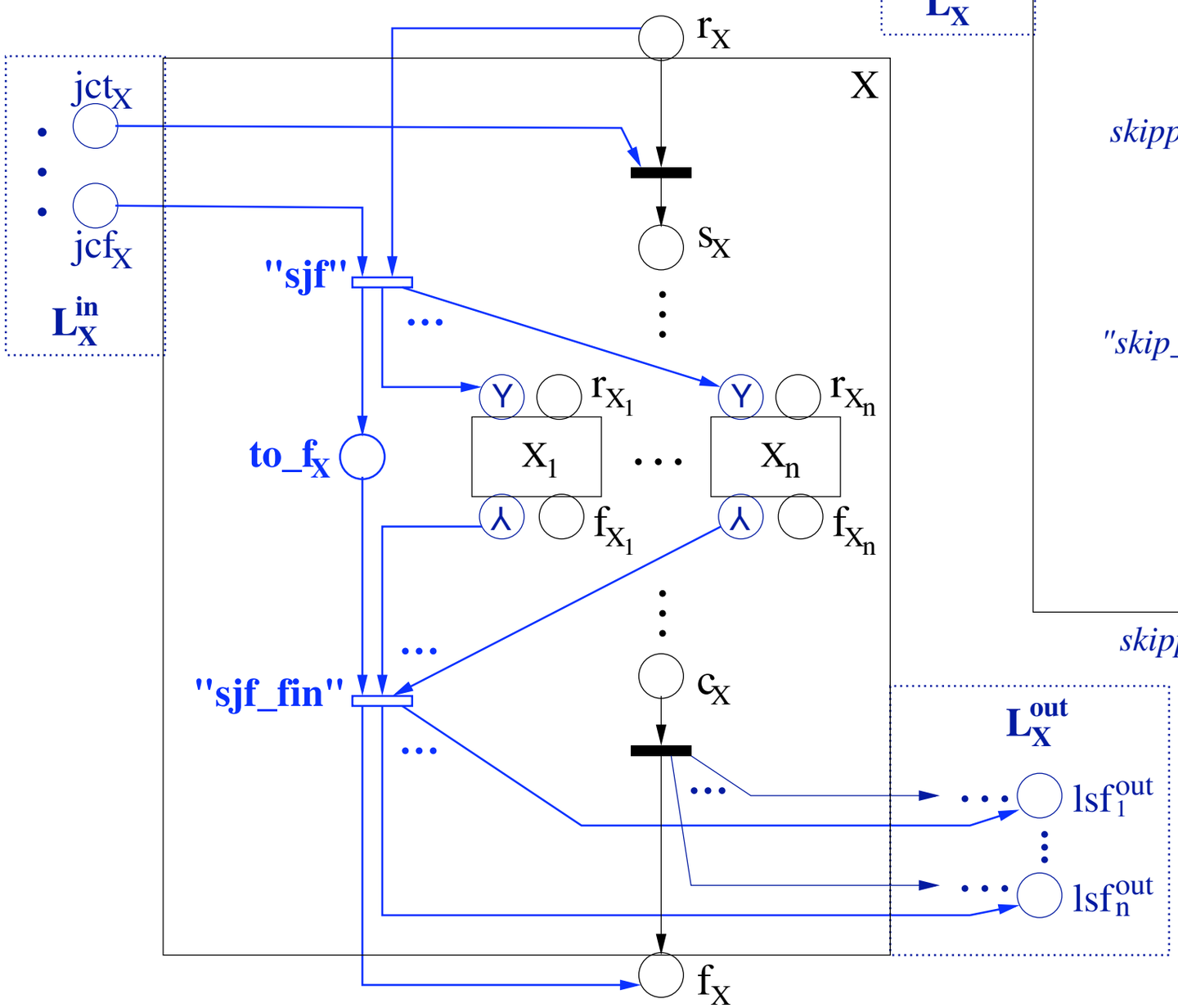


(a) normal behaviour

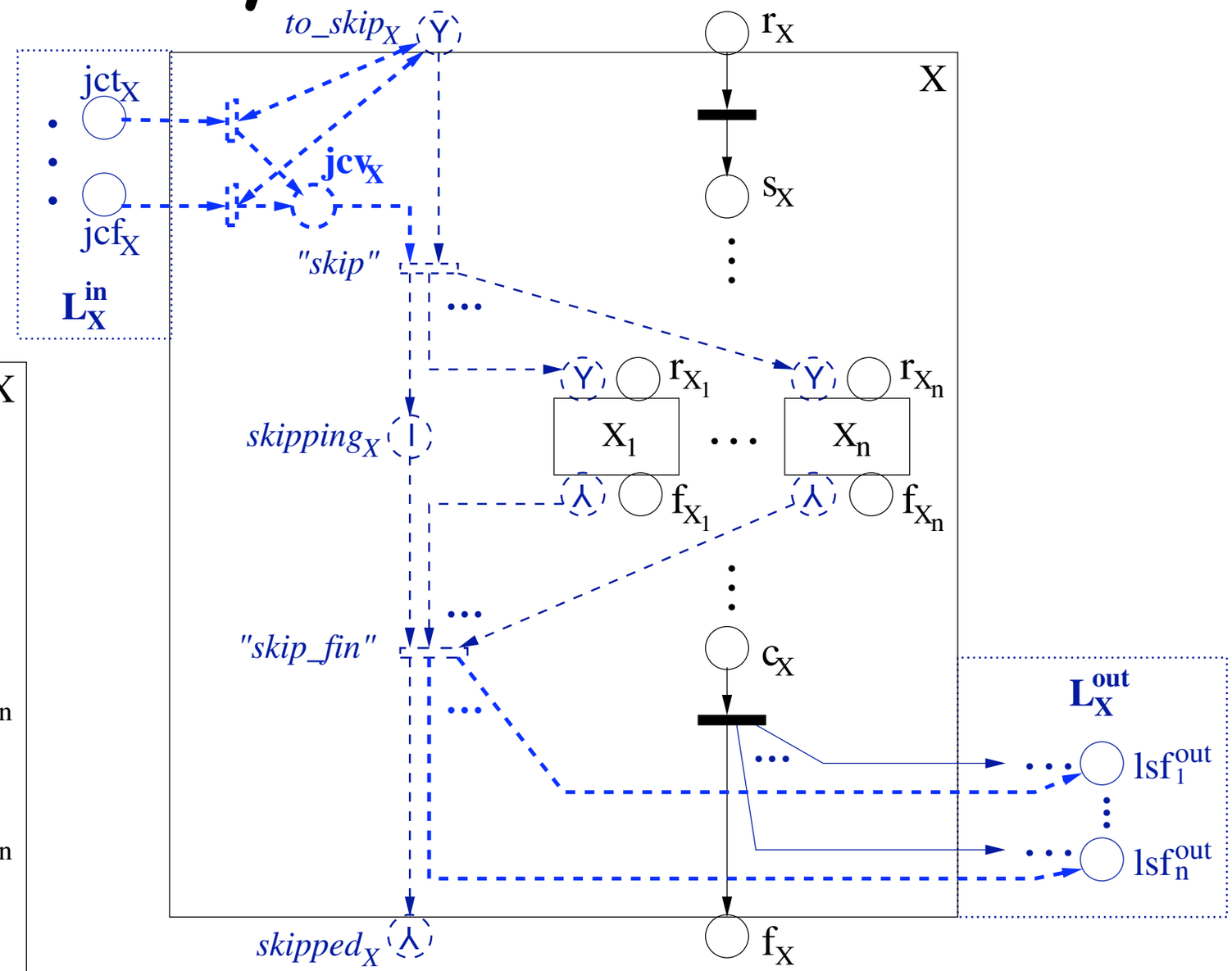


(b) skipping behaviour

Non-sequence activity with control links

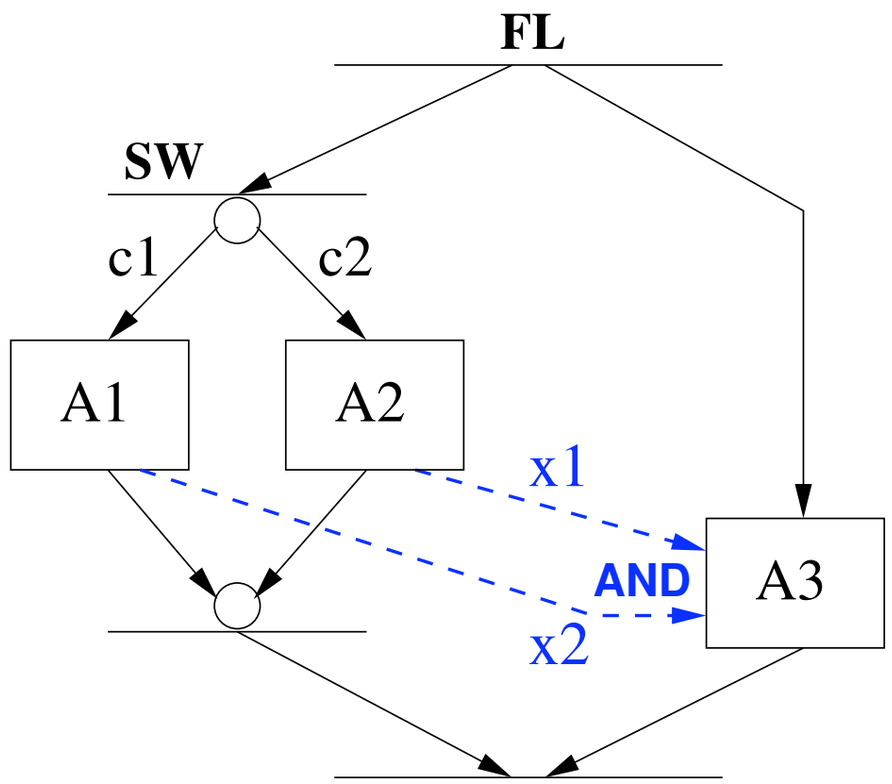


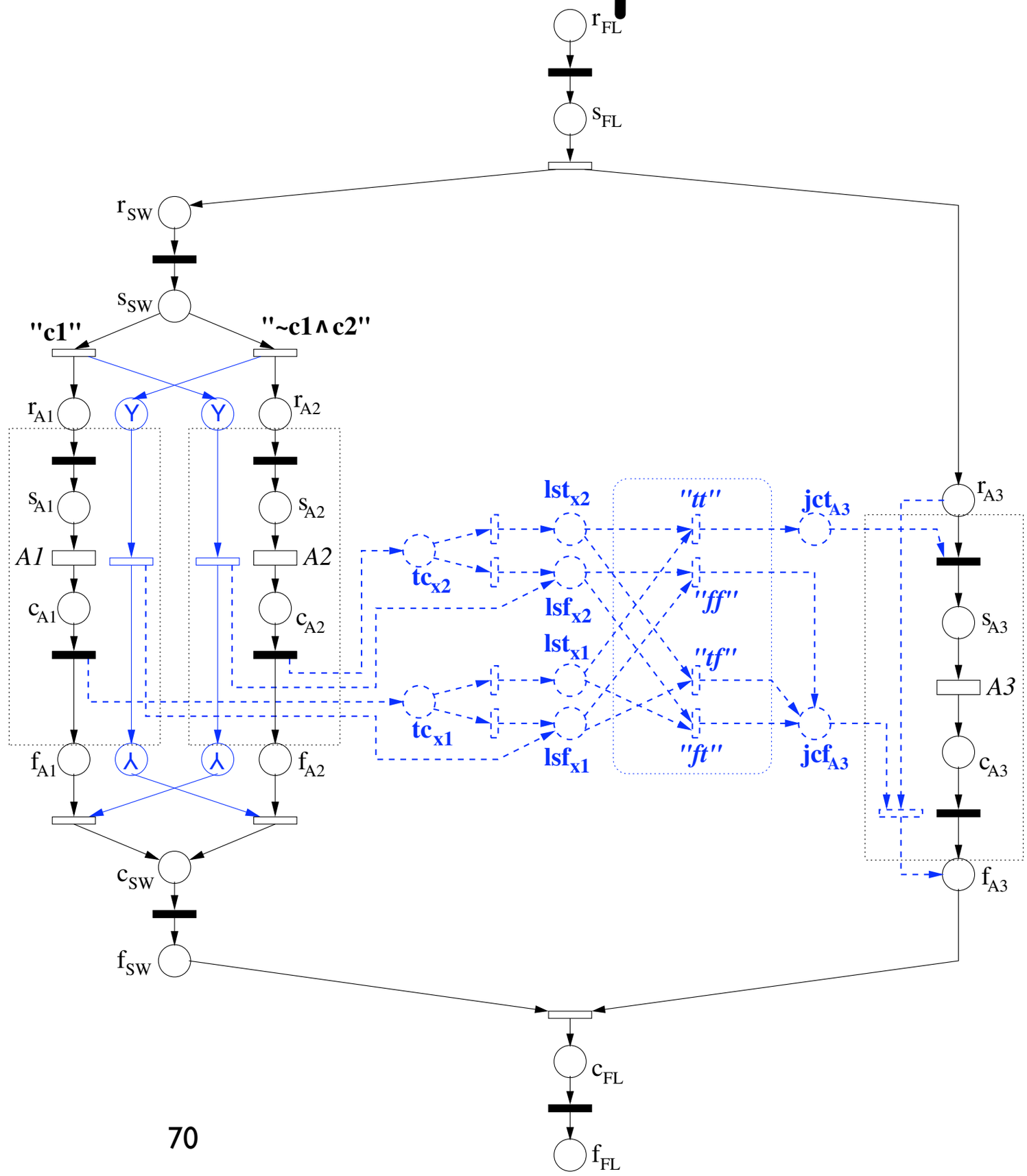
(a) normal behaviour



(b) skipping behaviour

The previous example





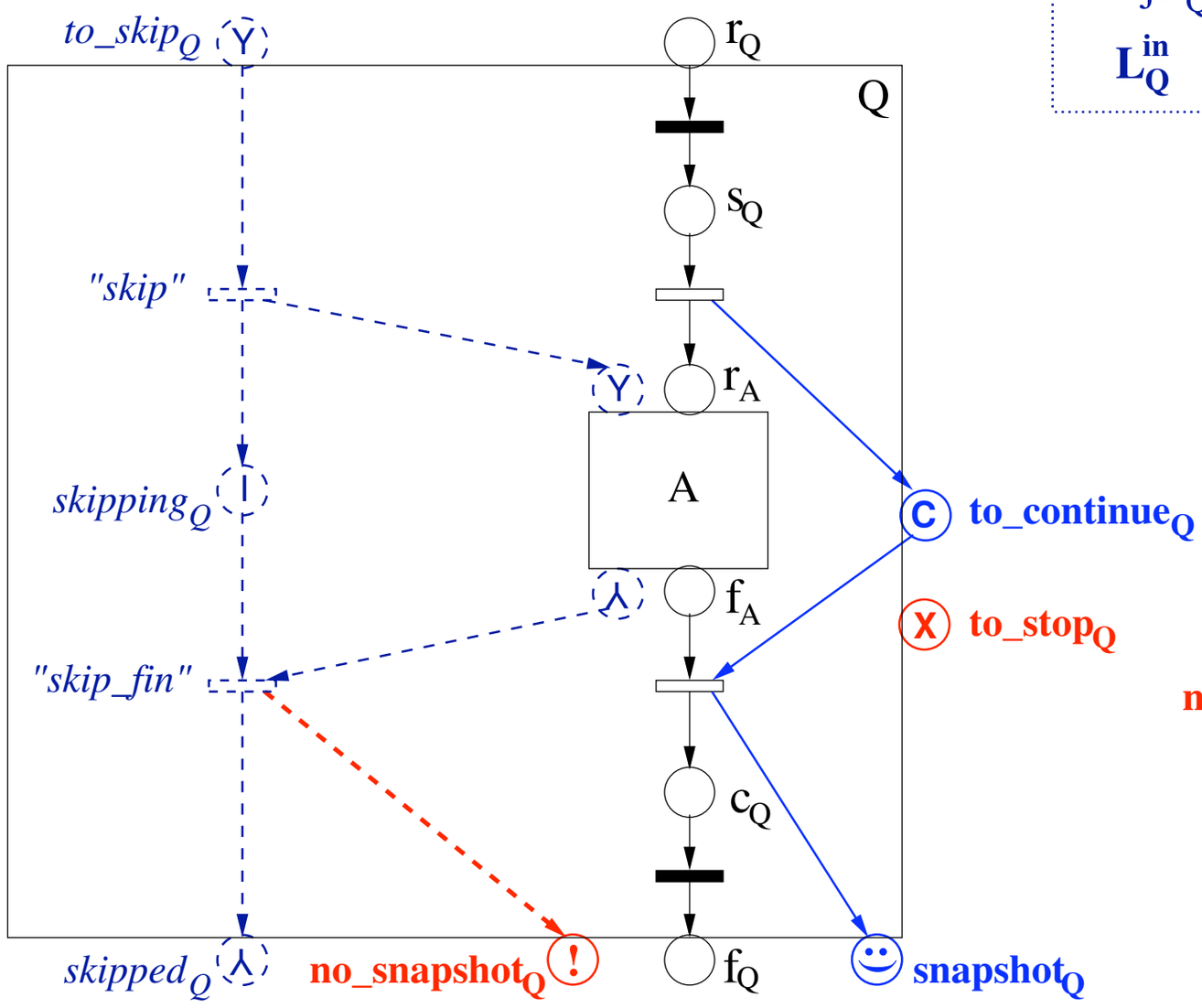
Scope

Remind that a scope has a primary activity, and optionally:
a set of fault handlers,
a set of event handlers, and
one compensation handler.

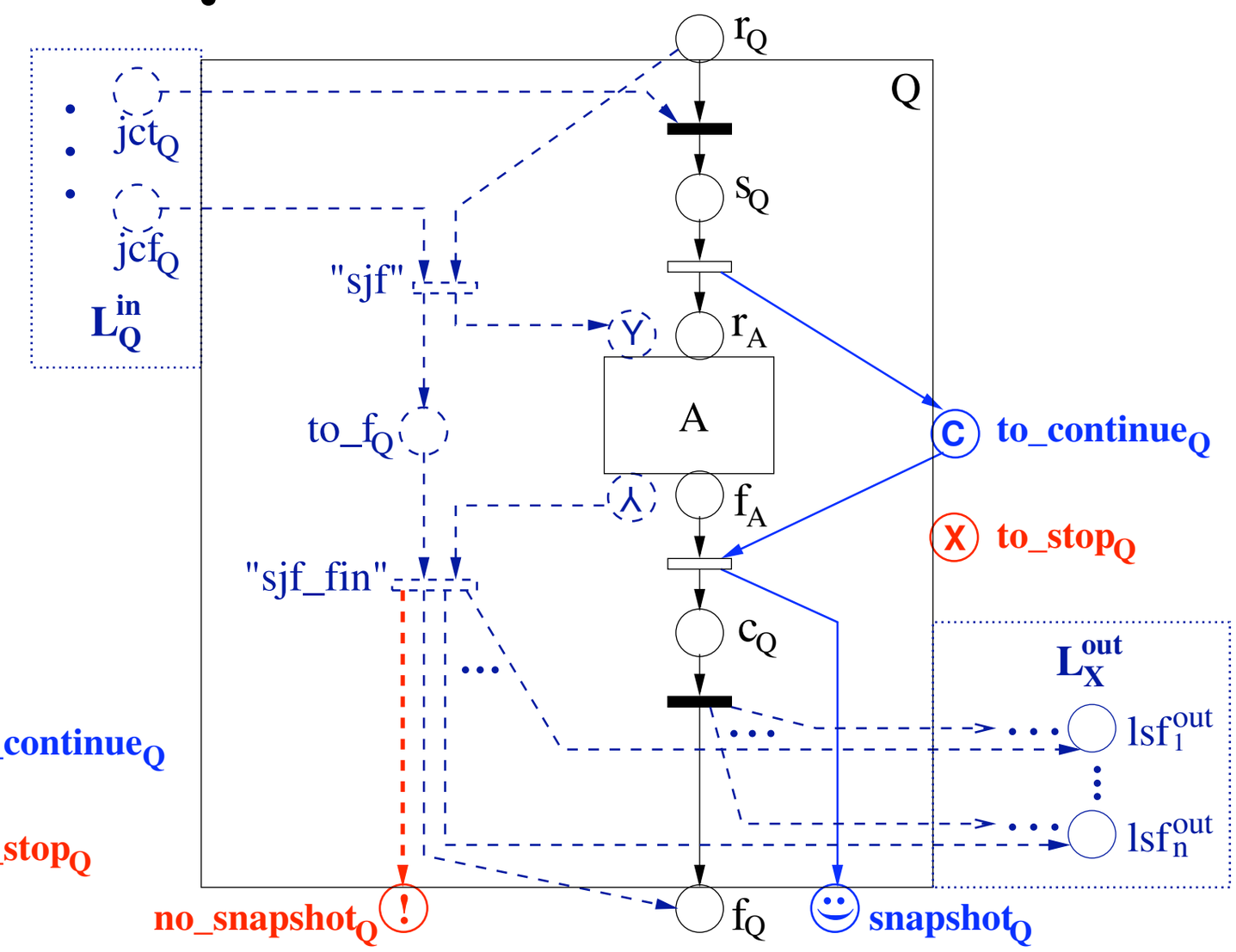
To deal with them, four “flags” are attached to a scope:
`to_continue` (no exception, execution is in progress)
`to_stop` (an error occurred, activities need to stop)
`snapshot` (successfully completed, uncompensated)
`no_snapshot` (no compensation needed)

In the following, we just sketch the handling of faults

Scope



(a) skipping behaviour



(b) suppressing join failure

Faults

BPEL defines three kinds of faults:

application faults (also **service faults**)
generated by invoked services

process-defined faults
generated by a **`<throw>`** activity

system faults
generated by the process engine, such as join failures

“it is never possible to run more than one fault handler for the same scope, under any circumstances”

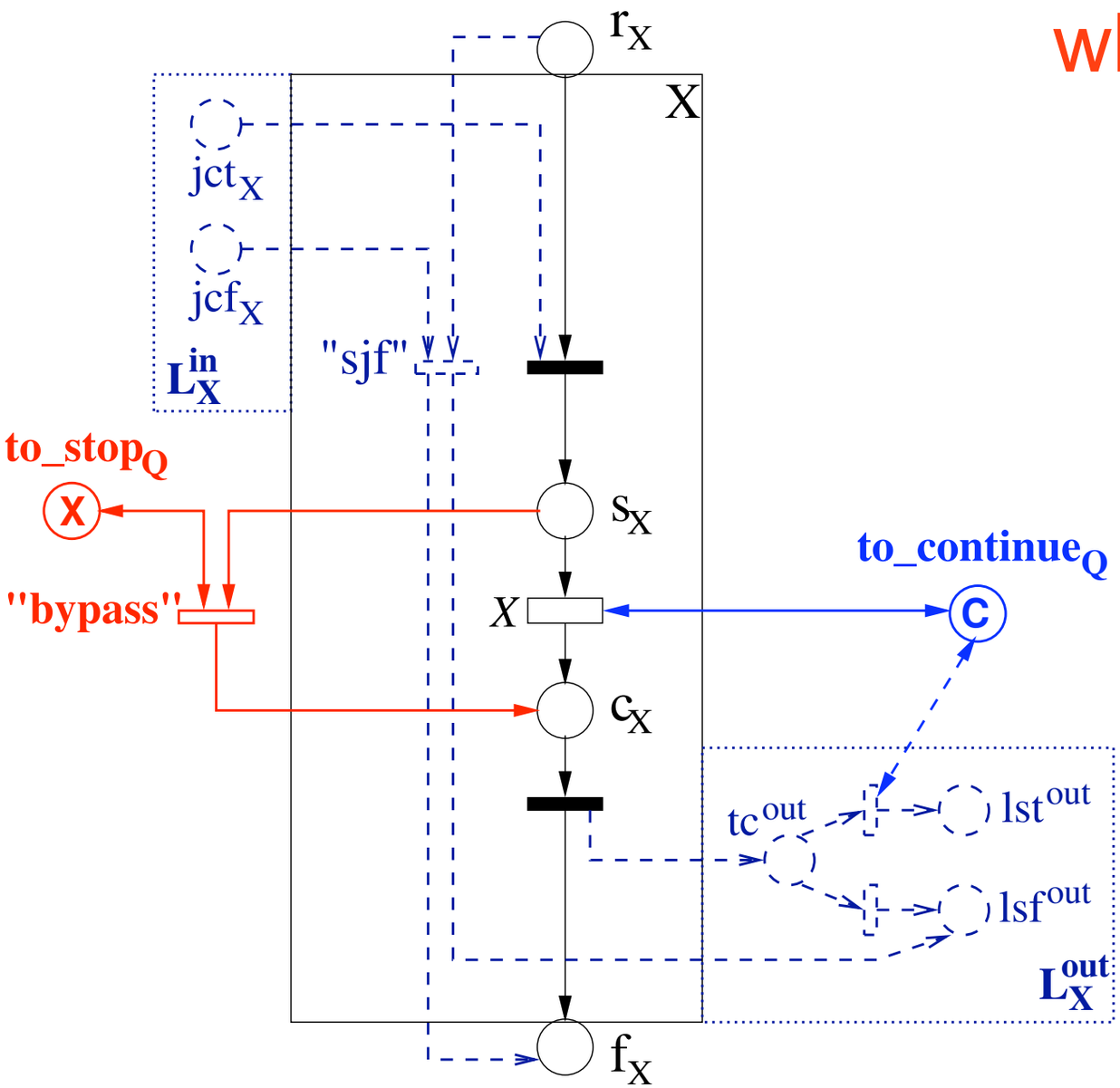
Some assumptions

It is not shown entirely here, but the scope is encoded in such a way that when the token is in **`to_stop`** (instead of **`to_continue`**)
all active activities in the scope are by-passed: they will be terminated reaching their “finished” state

We do not show the case of a fault occurred during the faulty mode of a scope (it is handled by the parent scope)

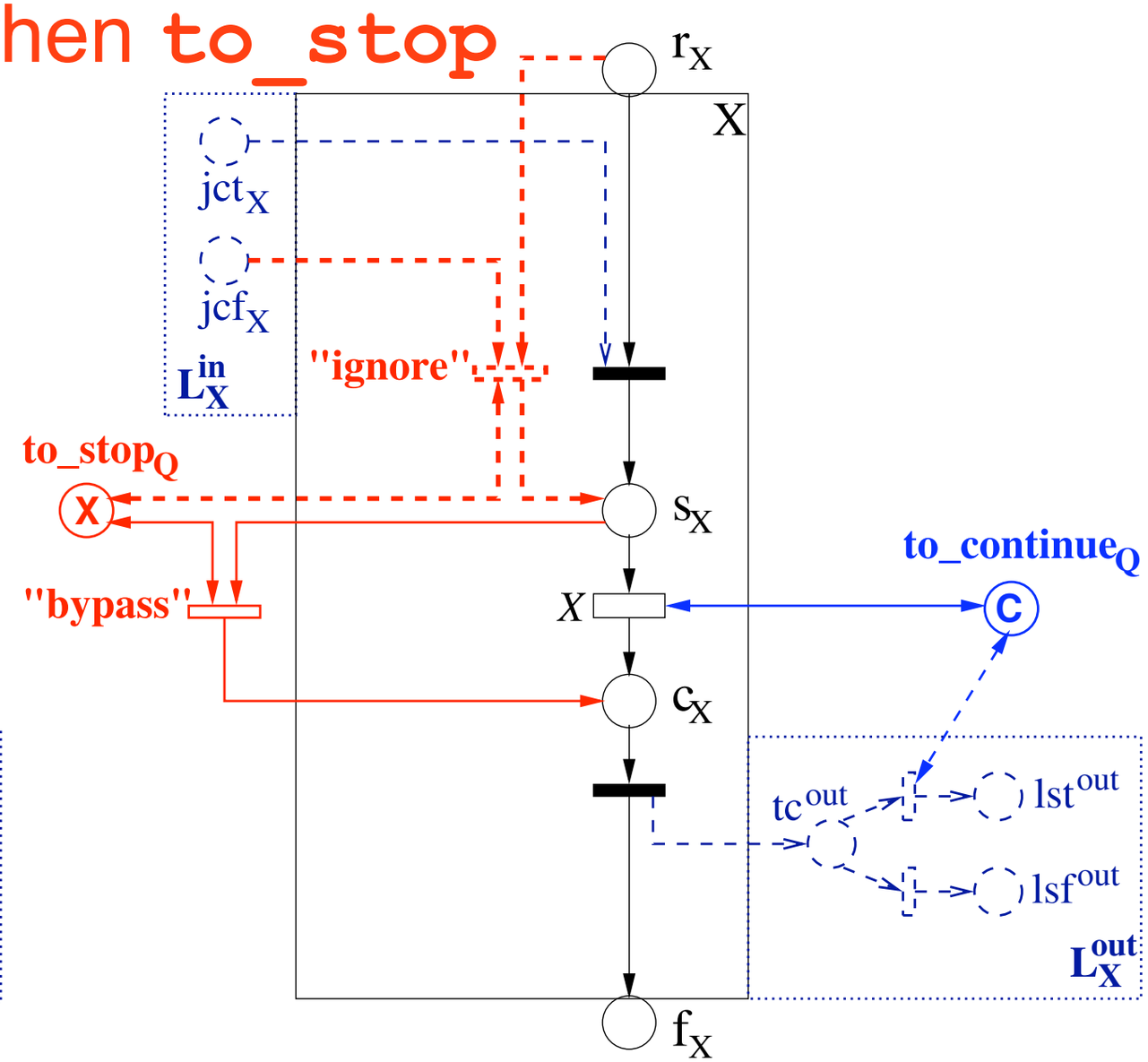
Control links are only allowed to leave the boundary of a fault handler: we do not show dead-path elimination for them

By-pass a basic activity



(a) suppressJoinFailure = yes

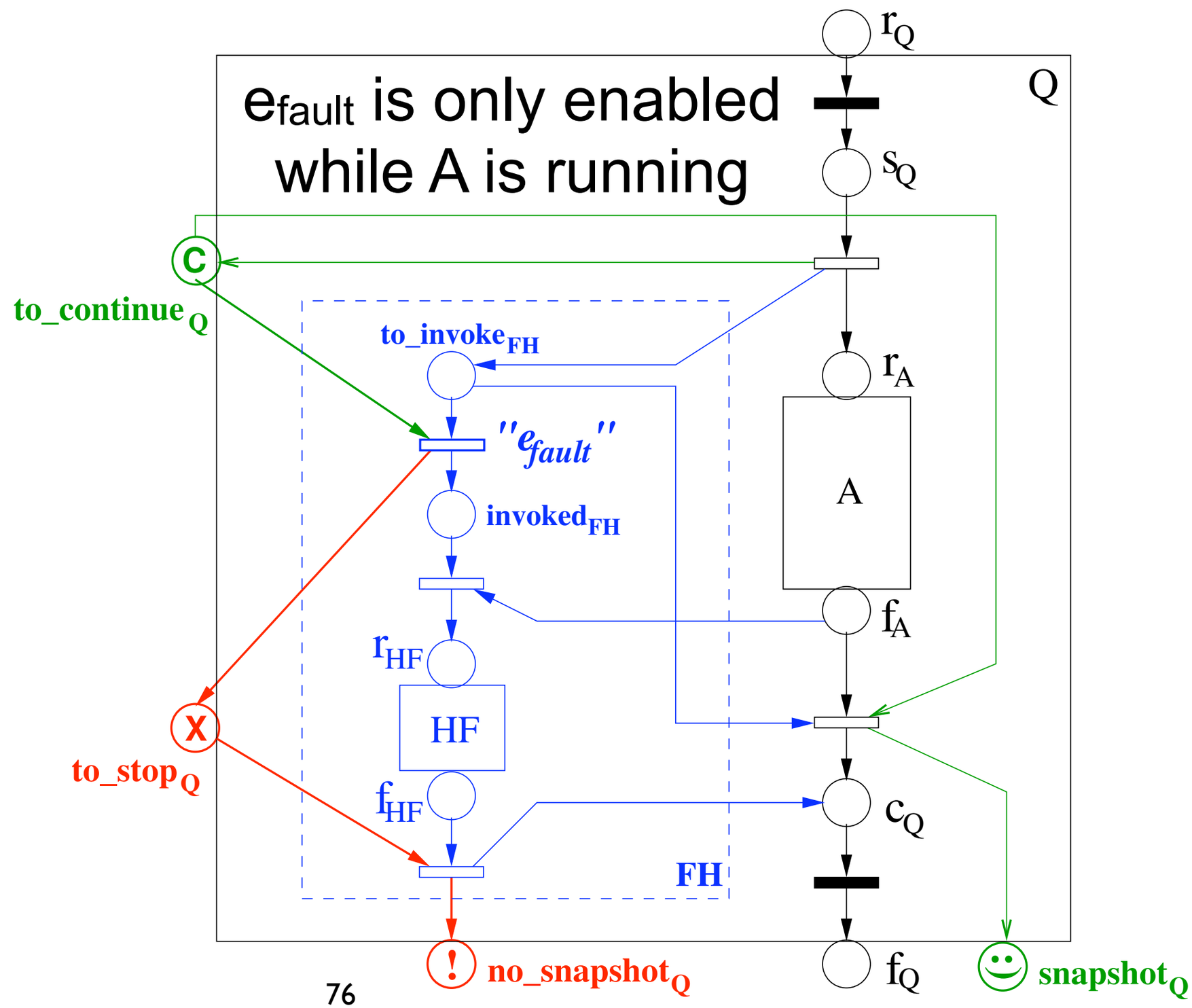
ignore join-failure
when `to_stop`



(b) suppressJoinFailure = no

A fault handler (general case)

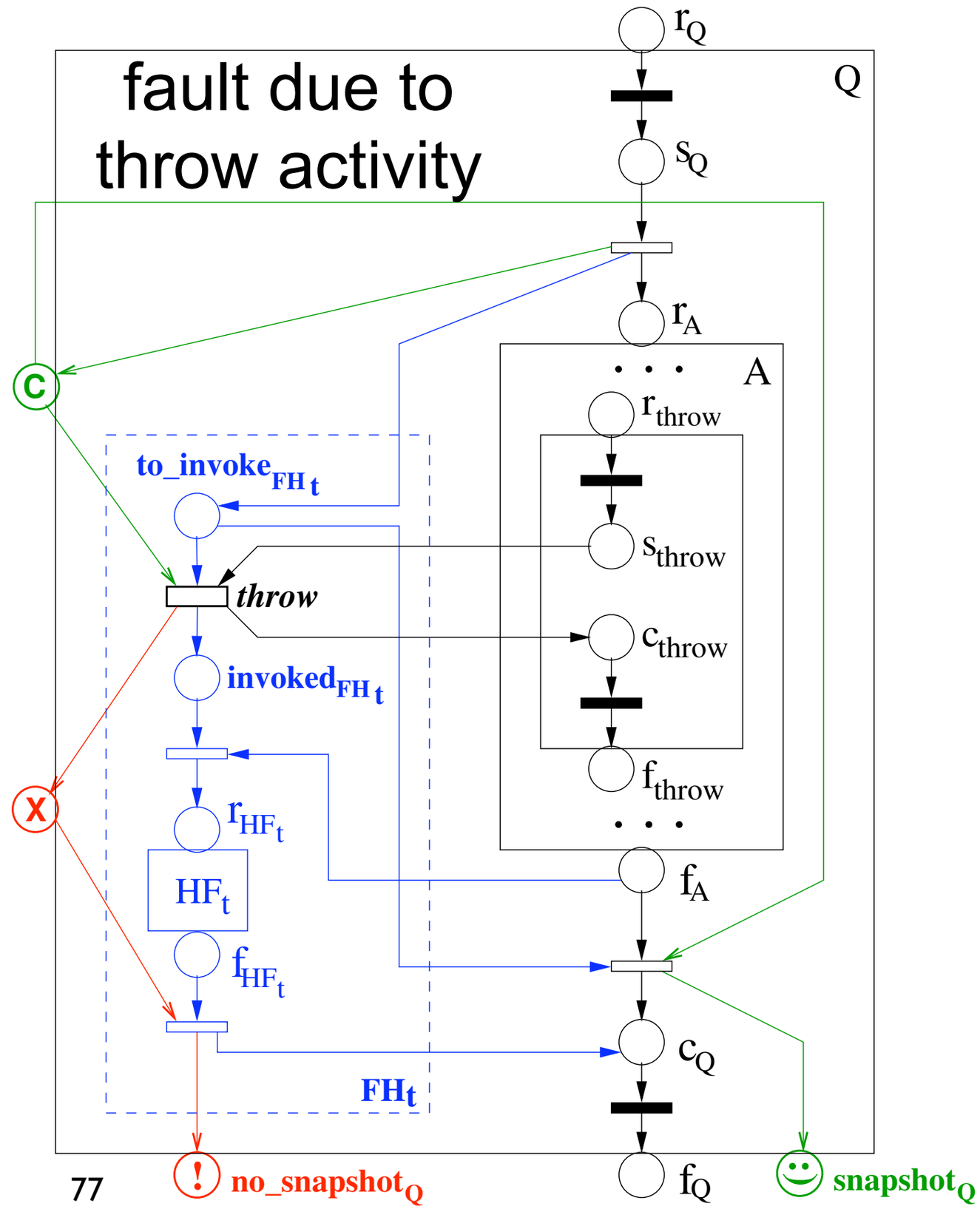
```
<scope name="Q">
  <faultHandlers>
    <catch faultName="fault">
      activityHF
    </catch>
  </faultHandlers>
  activityA
</scope>
```



Example: process-defined fault

```
<scope name="Q">
  <faultHandlers>
    <catch faultName="fault">
      activityHF_t
    </catch>
  </faultHandlers>
  <activityA>
      ⋮
    <throw faultName="fault"/>
      ⋮
  </activityA>
</scope>
```

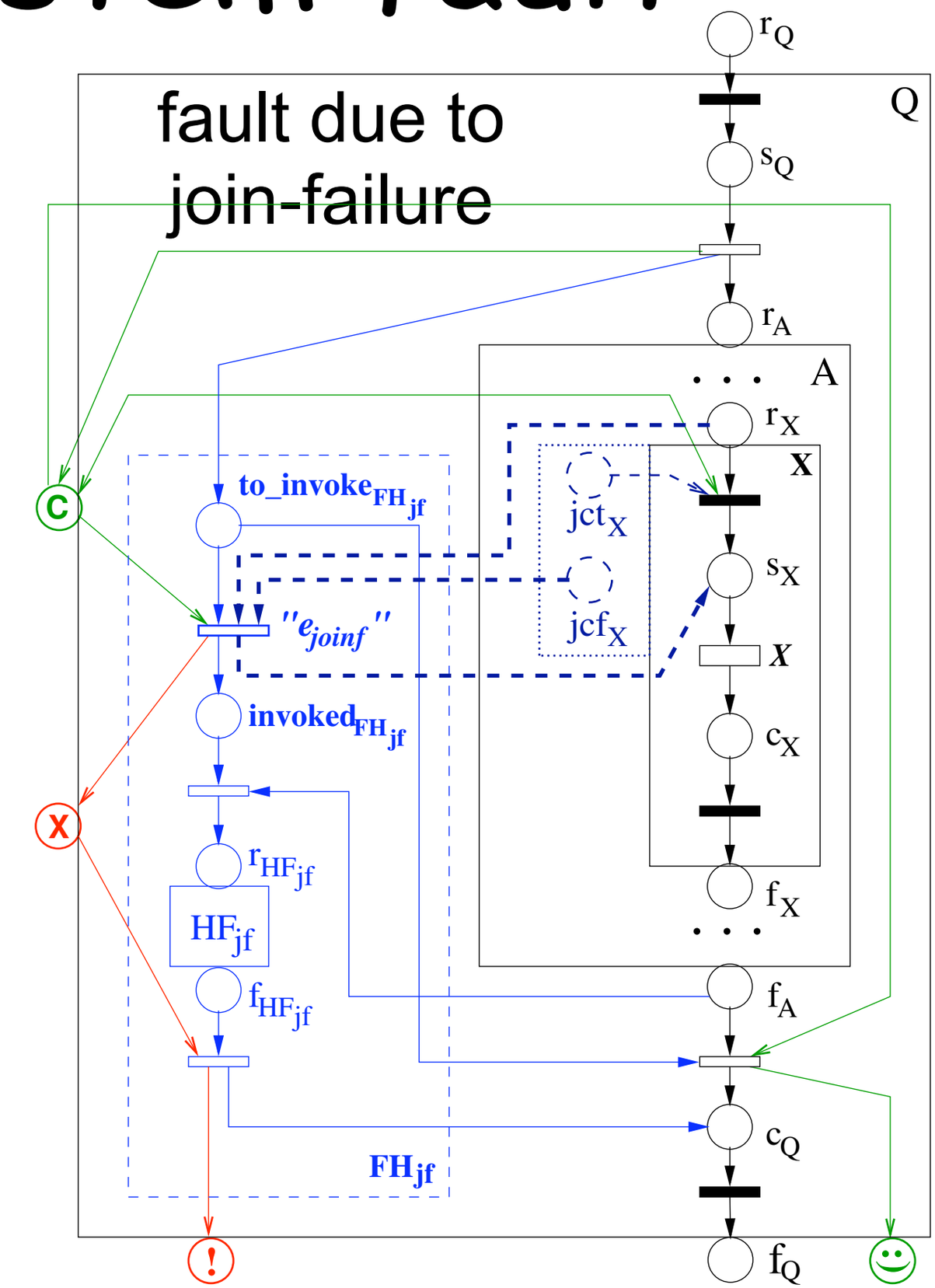
[note] *The above throw activity in A is directly enclosed in scope Q.*



Example: system fault

```
<scope name="Q">
  <faultHandlers>
    <catch faultname="bpws: joinFailure">
        activity HF_jf
    </catch>
  </faultHandlers>
  <activityA>
        ⋮
    <activityX suppressJoinFailure="no">
      <targets>
        <joinCondition>
            •••
        </joinCondition>
        <target linkname= ••• >
      </targets>
    </activityX>
        ⋮
  </activityA>
</scope>
```

[note] *X is directly enclosed in scope Q.*



Expressiveness...
not in the formal sense
(w.r.t. WF patterns)

Basic Control Flow Patterns

- Pattern 1 (Sequence)
- Pattern 2 (Parallel Split)
- Pattern 3 (Synchronization)
- Pattern 4 (Exclusive Choice)
- Pattern 5 (Simple Merge)

Advanced Branching and Synchronization Patterns

- Pattern 6 (Multi-choice)
- Pattern 7 (Synchronizing Merge)
- Pattern 8 (Multi-merge)
- Pattern 9 (Discriminator)

Structural Patterns

- Pattern 10 (Arbitrary Cycles)
- Pattern 11 (Implicit Termination)

Cancellation Patterns

- Pattern 19 (Cancel Activity)
- Pattern 20 (Cancel Case)

State-based Patterns

- Pattern 16 (Deferred Choice)
- Pattern 17 (Interleaved Parallel Routing)
- Pattern 18 (Milestone)

Patterns involving Multiple Instances

- Pattern 12 (Multiple Instances Without Synchronization)
- Pattern 13 (Multiple Instances With a Priori Design Time Knowledge)
- Pattern 14 (Multiple Instances With a Priori Runtime Knowledge)
- Pattern 15 (Multiple Instances Without a Priori Runtime Knowledge)

pattern	*standard*				
	BPEL	XLANG	WSFL	BPML	WSCI
Sequence	+	+	+	+	+
Parallel Split	+	+	+	+	+
Synchronization	+	+	+	+	+
Exclusive Choice	+	+	+	+	+
Simple Merge	+	+	+	+	+
Multi Choice	+	−	+	−	−
Synchronizing Merge	+	−	+	−	−
Multi Merge	−	−	−	+/−	+/−
Discriminator	−	−	−	−	−
Arbitrary Cycles	−	−	−	−	−
Implicit Termination	+	−	+	+	+
MI without Synchronization	+	+	+	+	+
MI with a Priori Design Time Knowledge	+	+	+	+	+
MI with a Priori Runtime Knowledge	−	−	−	−	−
MI without a Priori Runtime Knowledge	−	−	−	−	−
Deferred Choice	+	+	−	+	+
Interleaved Parallel Routing	+/−	−	−	−	−
Milestone	−	−	−	−	−
Cancel Activity	+	+	+	+	+
Cancel Case	+	+	+	+	+