Static Analysis of Active XML Systems

S. Abiteboul — INRIA
L. Segoufin — INRIA
V. Vianu — UCSD

Broader context:
Verification of data-driven systems

- growing area at the intersection of

  Databases and Computer-Aided Verification
- some promising theory and implementation
- potential for significant practical impact

Applications centered around a database

- Data-driven Web services
- E-commerce
- E-government
- Business process support
- Scientific applications

Complex, prone to costly bugs:

need for verification!

How to verify

General-purpose software verification techniques

- model checking

  usually requires finite-state abstraction

- theorem proving

  incomplete, requires expert user feedback

Unsatisfactory!

- Recent work: can do better by taking advantage of a proliferation of high-level specification tools

  WebML, Wave, Hilda: Web sites/services

  Siena (IBM): Business processes/artifacts

  Active XML (INRIA): XML services

- Ideal targets for verification

Good news: Can automatically verify significant classes of applications!

Active XML:
XML with embedded service calls

- integrates the XML and Web service paradigms
- controlled materialization of data

    keep dynamic data fresh

- implement evolving documents

Ongoing project in Serge's group

SIGMOD 2003, PODS 04/08/09, VLDB Journal,TODS

AXML for evolving documents

- Role of AXML service: support processing tasks

  loan applications, mail orders, tax forms, etc

- Evolution of a document reflects stages in carrying out the task
- Tasks are initiated by functions calls

  embedded function calls → sub-tasks

- Governed by workflow specified implicitly

  by constraints on document evolution

Outline

- Specific variant of AXML

  Guarded AXML (GAXML)

- Language for specifying temporal properties

  Tree-LTL

- Results: boundary of decidability of verification

Challenge: infinite-state system!

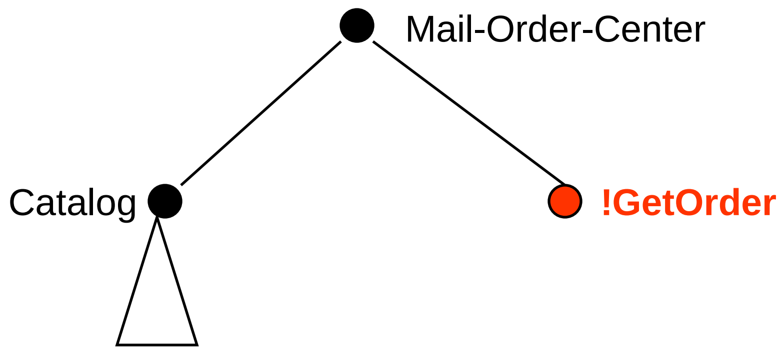
Guarded AXML

- guards used to control when a function is called and when the result is returned

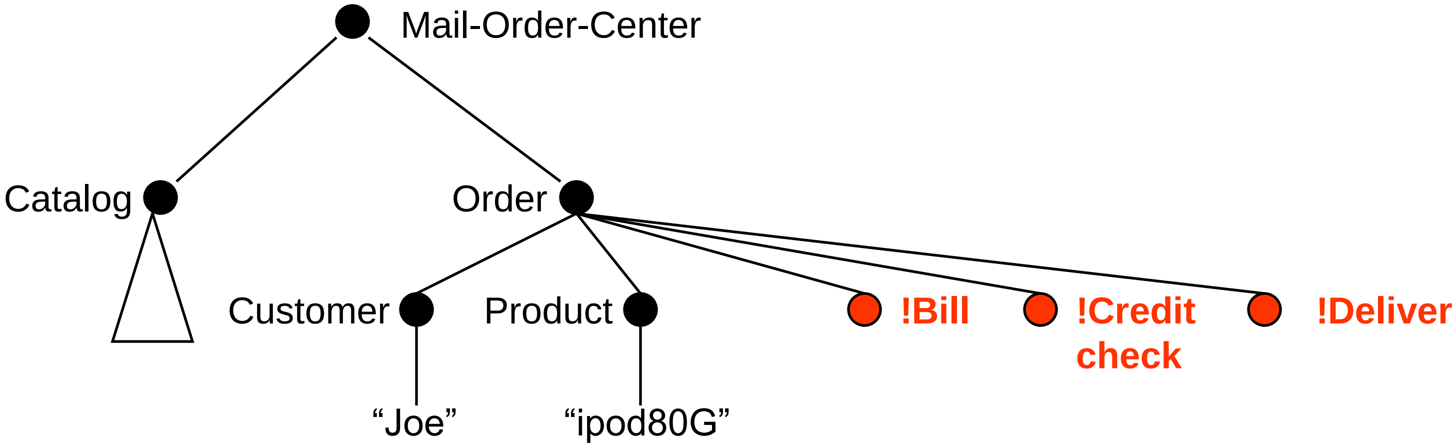
  powerful control mechanism

- data is passed as arguments to calls and as results of calls: defined by queries

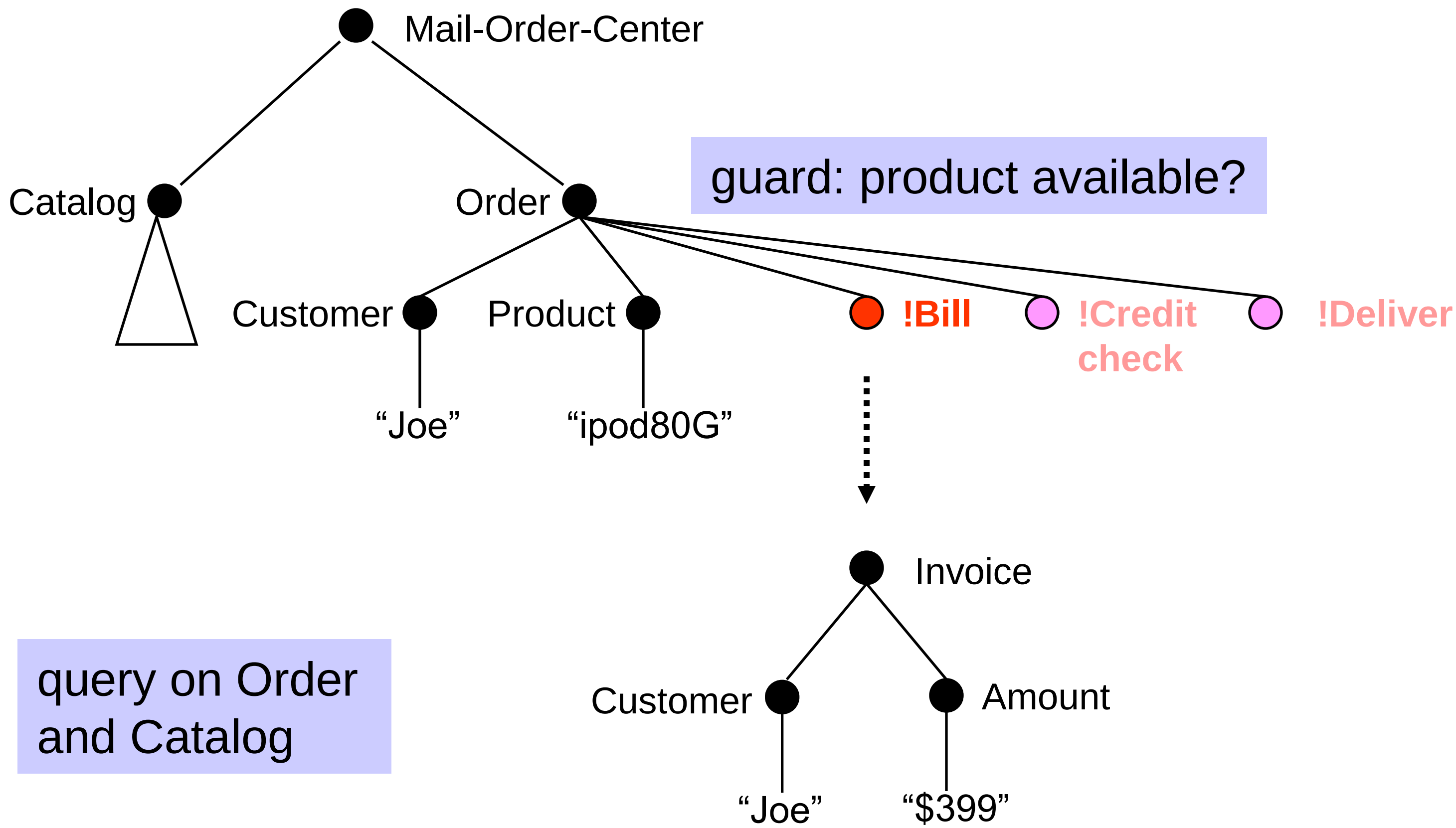
GAXML by example



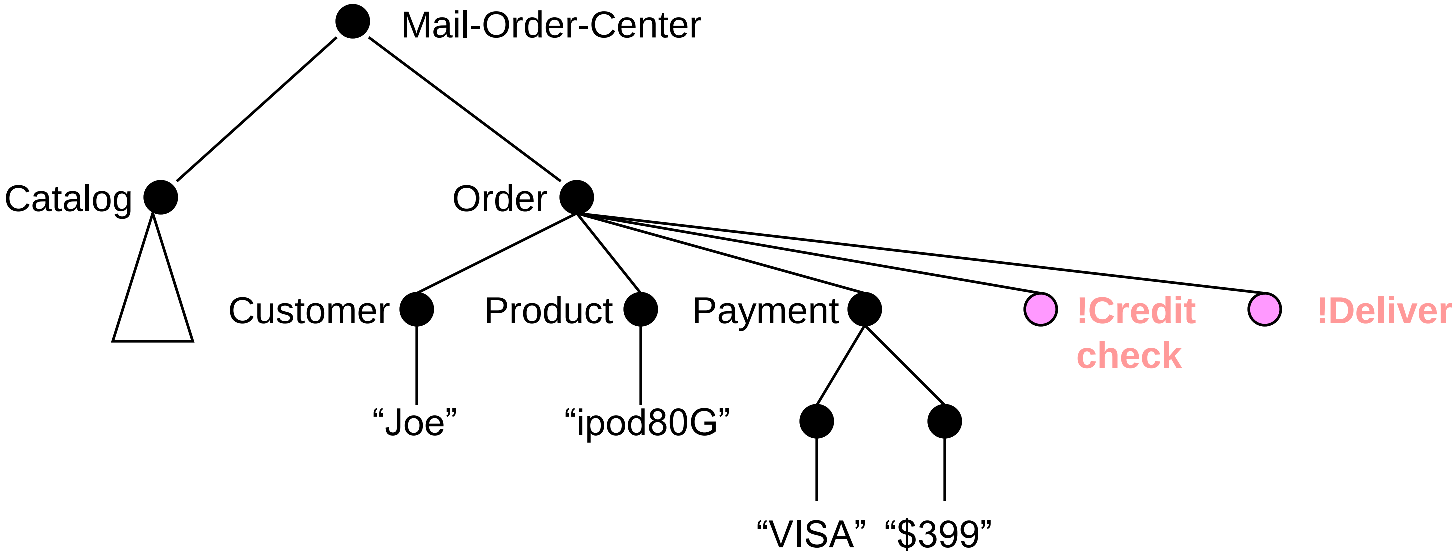
GAXML by example



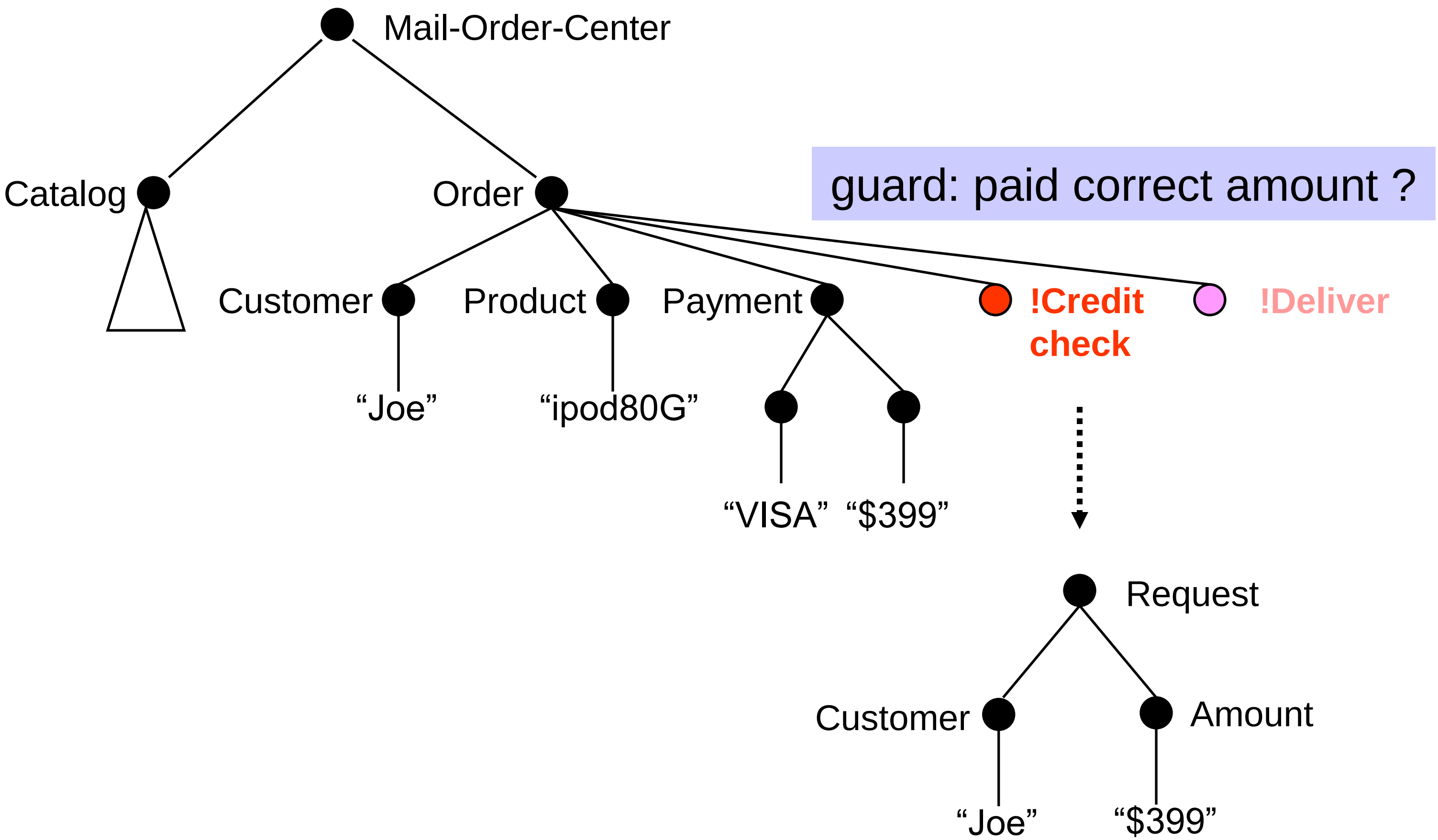
GAXML by example



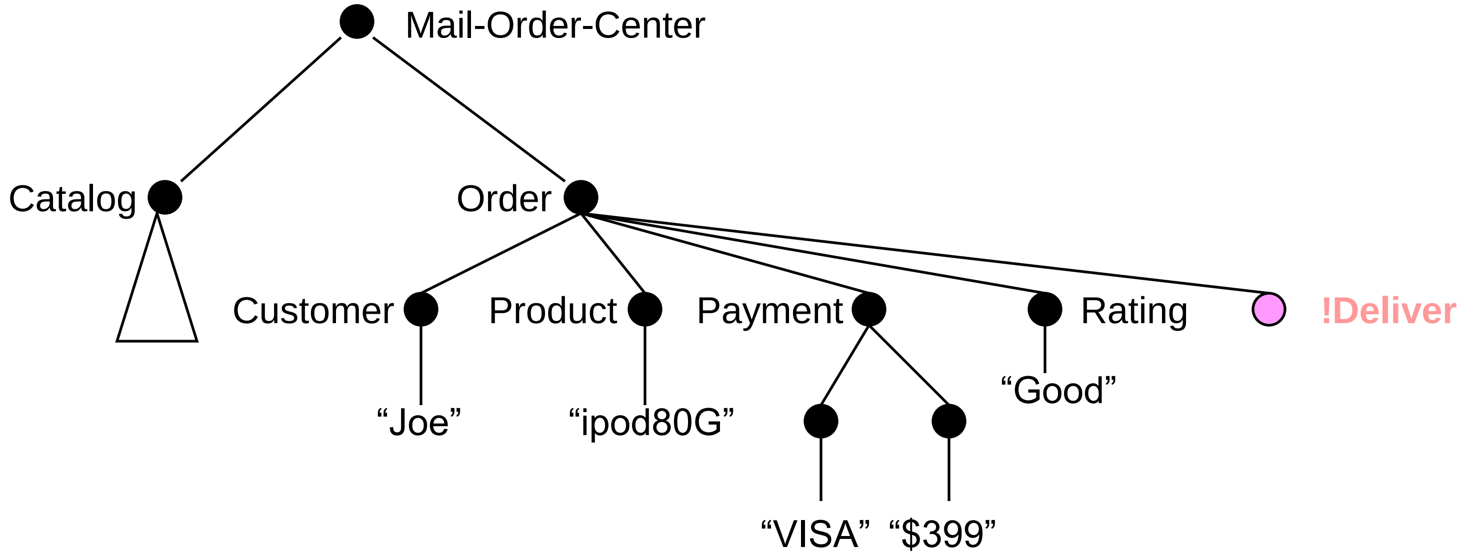
GAXML by example



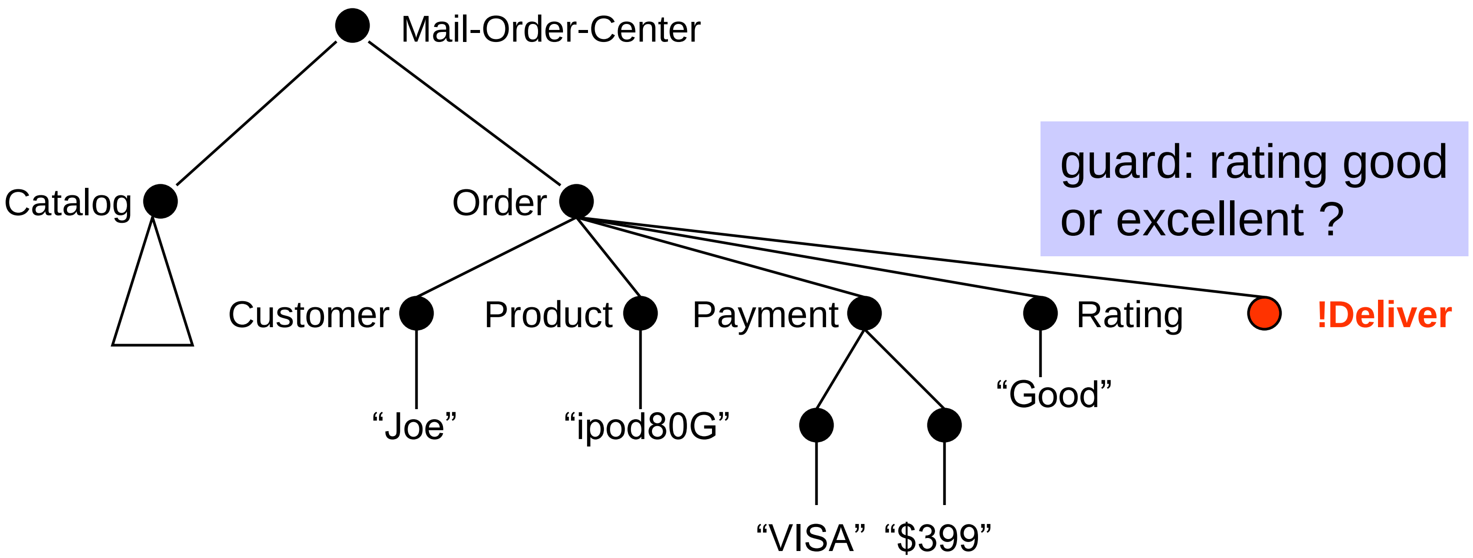
GAXML by example



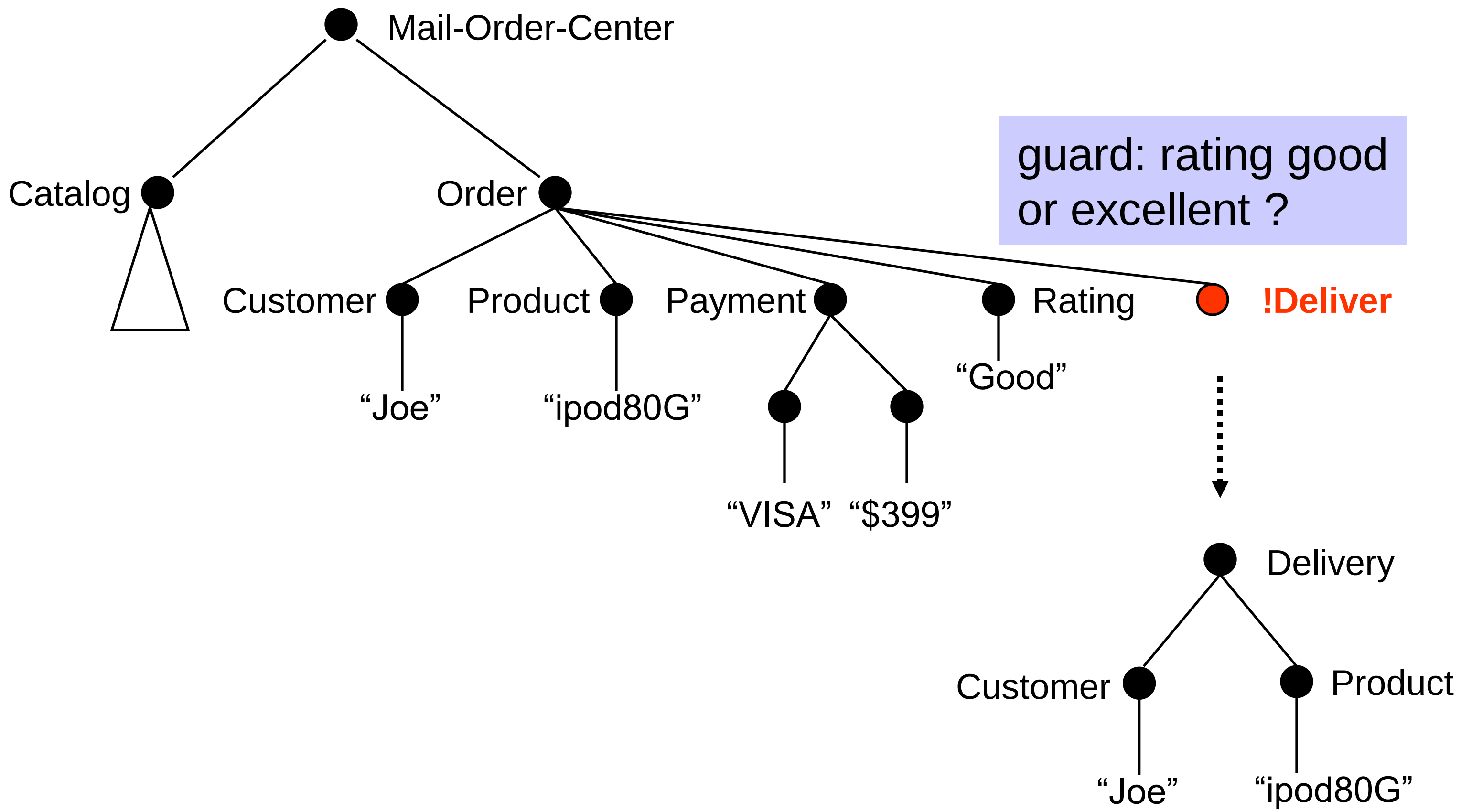
GAXML by example



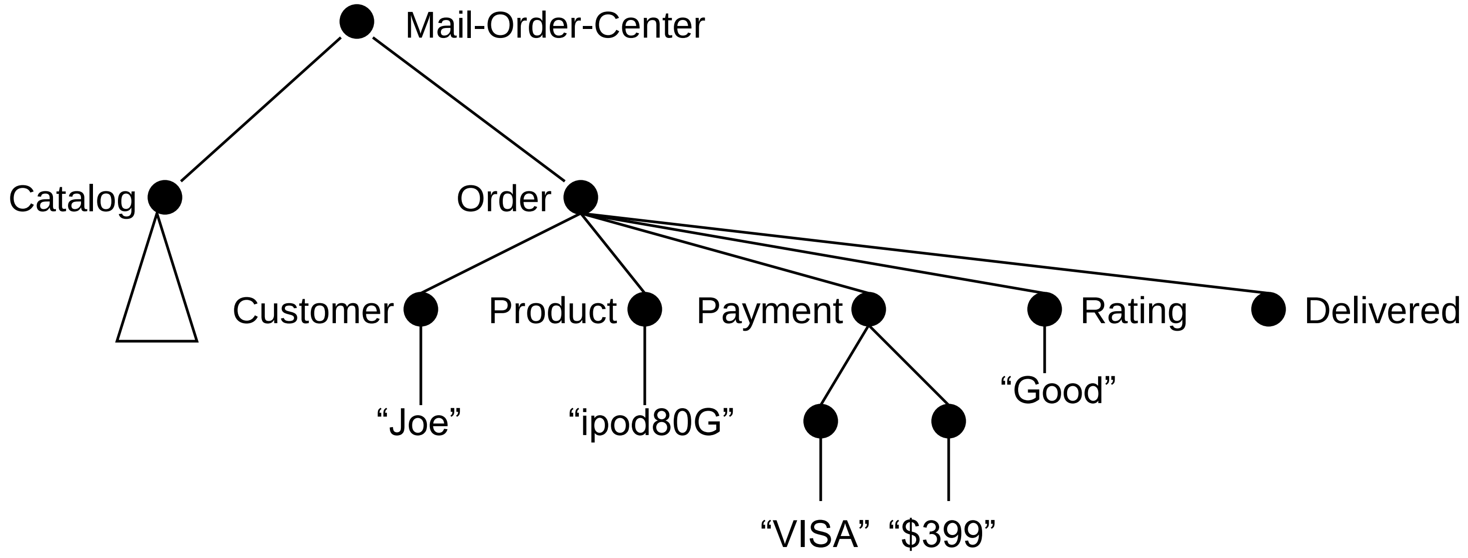
GAXML by example



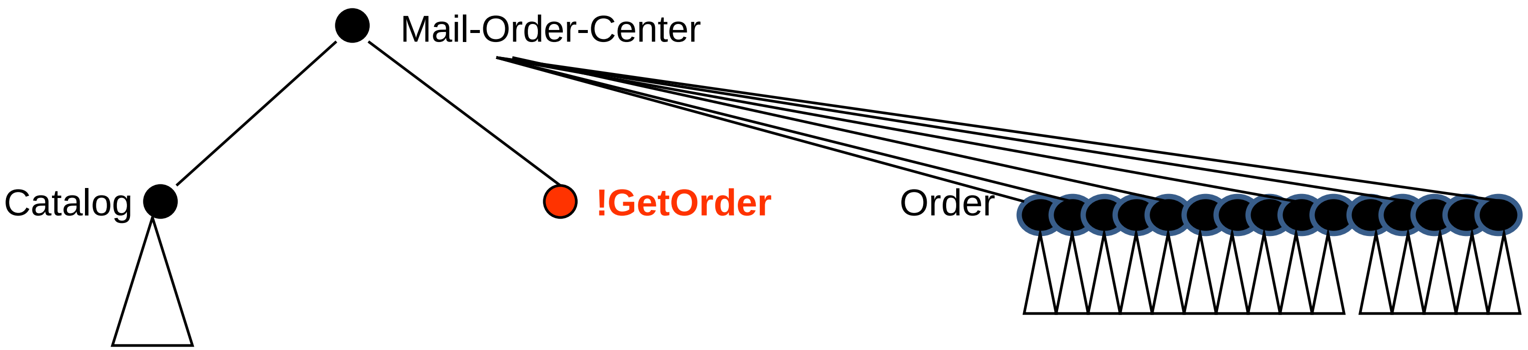
GAXML by example



GAXML by example



GAXML by example



More details

- GAXML documents: unordered trees
- Internal nodes are labeled by tags
- Leaves are labeled by tags, data values, or function symbols

  !f   call to f

  ?f   running call to f

Static constraints on documents

- DTDs

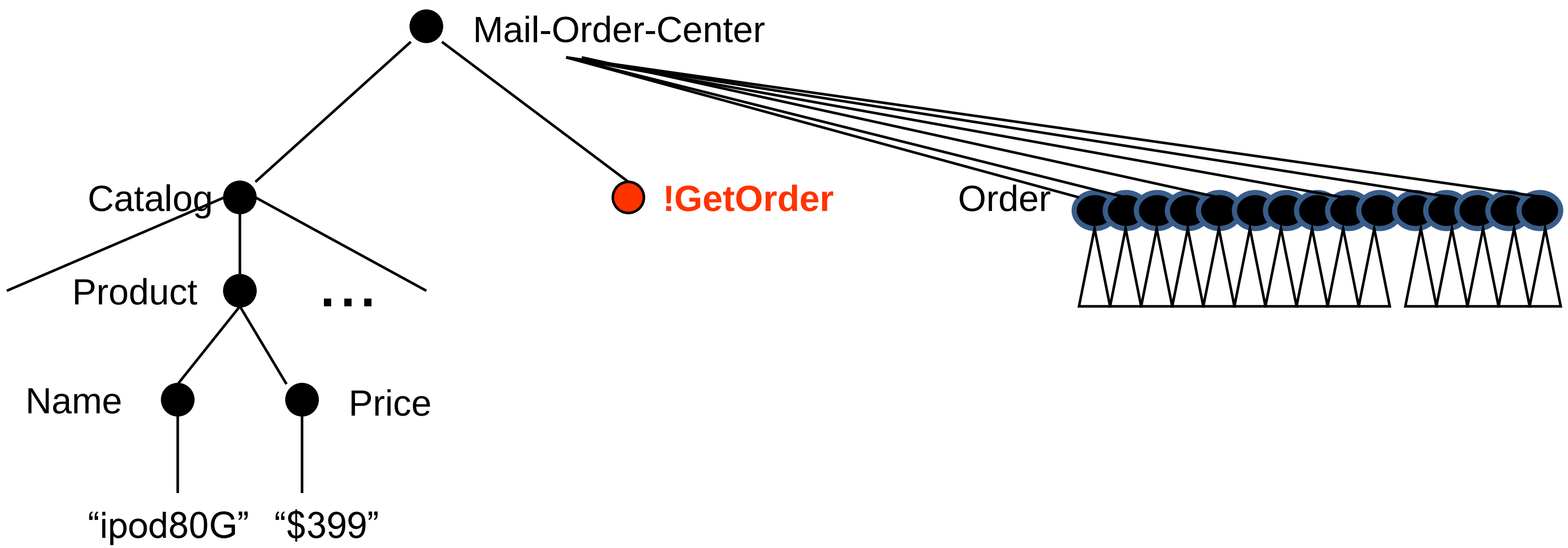
  restrict number of children with given tag, function symbol, or containing data values

- Data constraints

  Boolean combinations of tree patterns

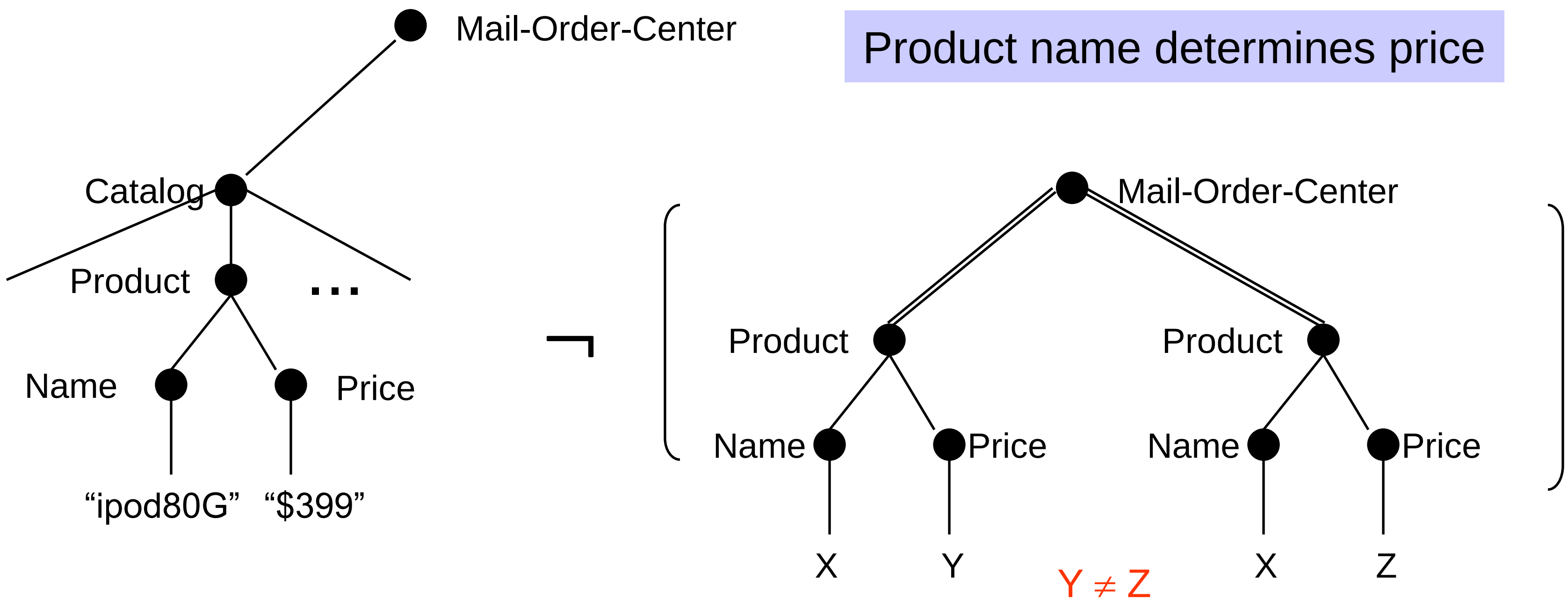
Static constraints on documents

- Example of data constraint



Static constraints on documents

- Example of data constraint



Function Guards and Queries

- call guards and return guards

  Boolean combinations of tree patterns
- argument and return queries

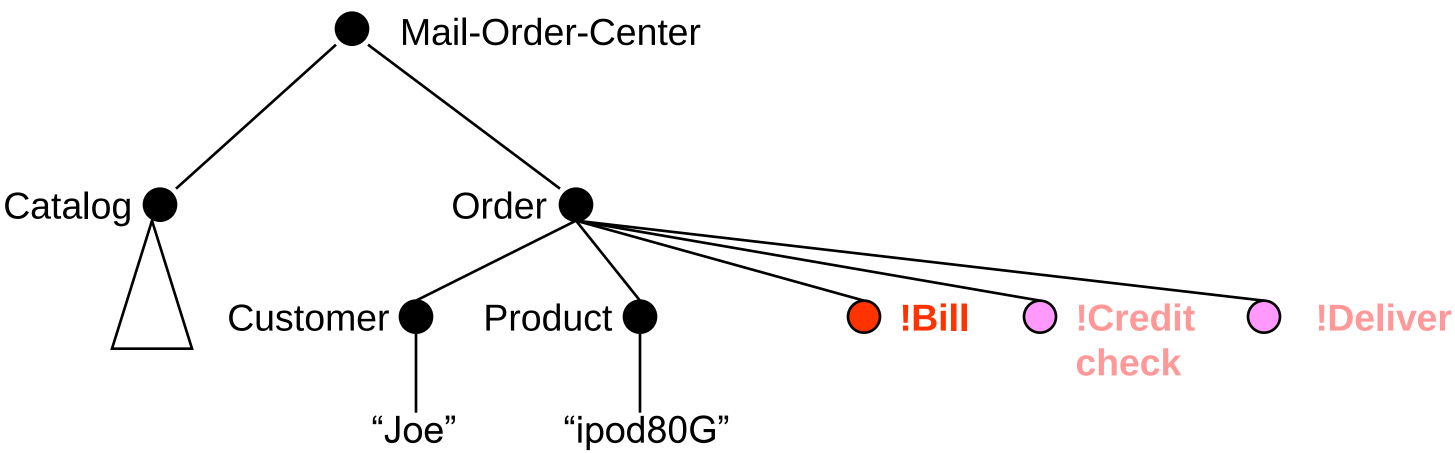
  similar to XML-QL

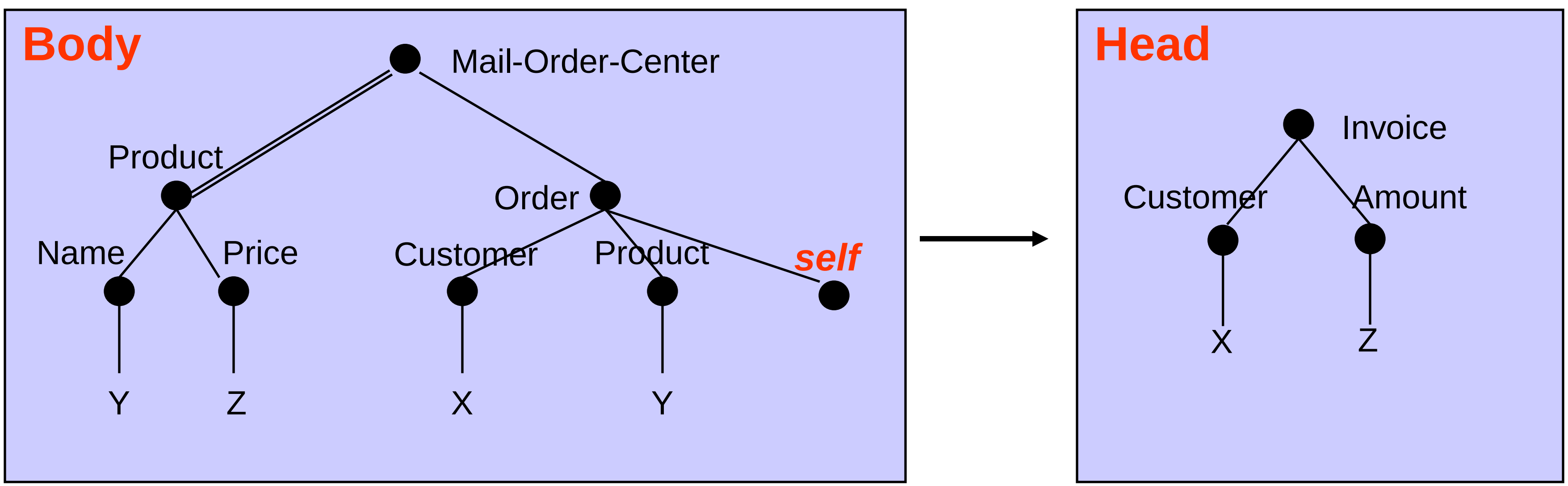
BODY → HEAD

collect variable bindings

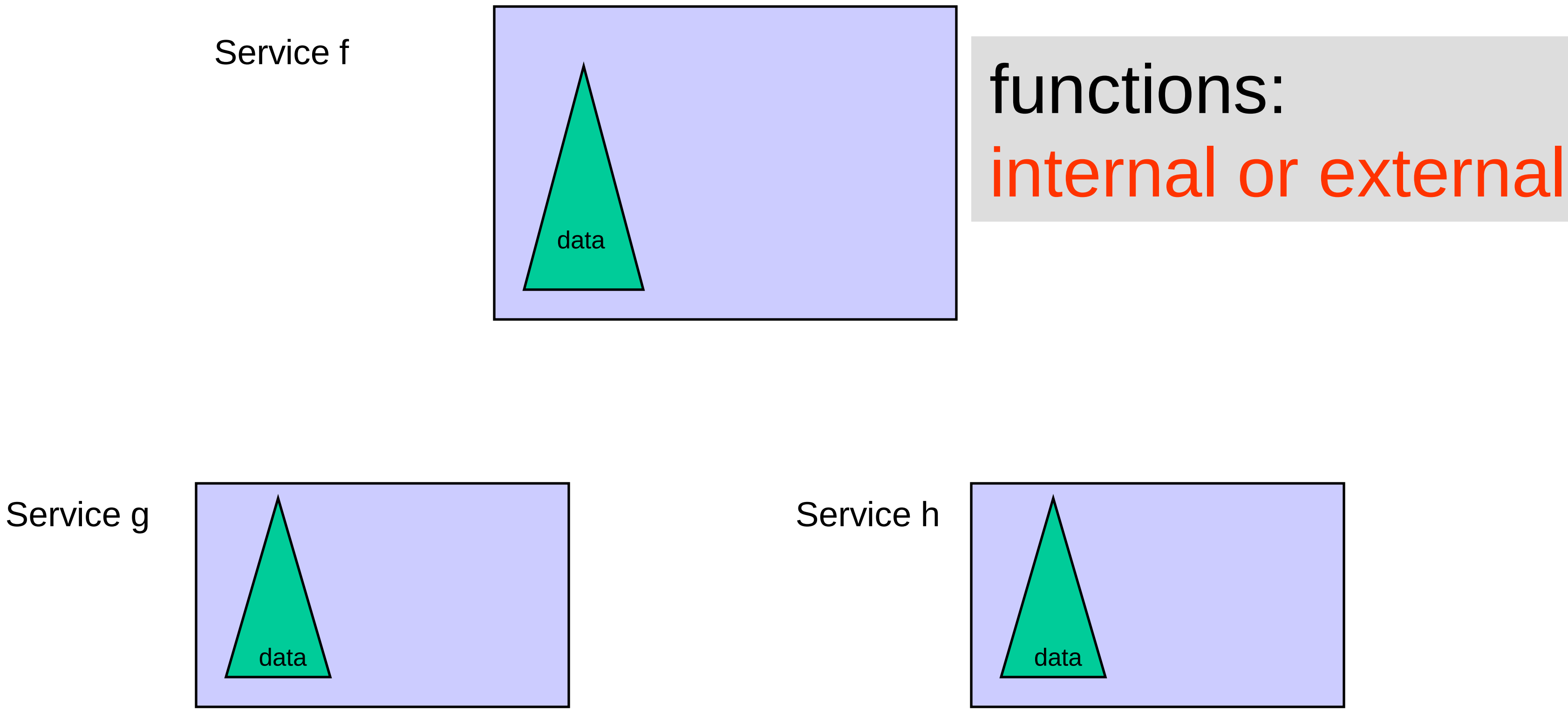
construct result

Example: argument query for **!Bill**



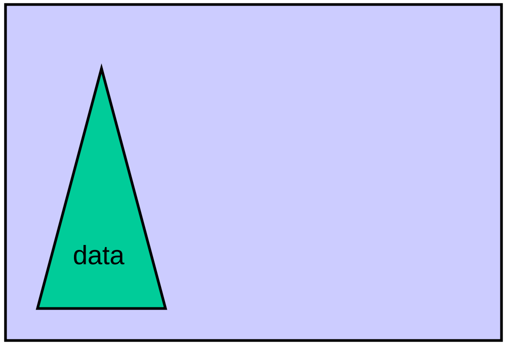


GAXML system: overall picture



GAXML system: overall picture

Service f



GAXML service specification:

- static constraints on trees

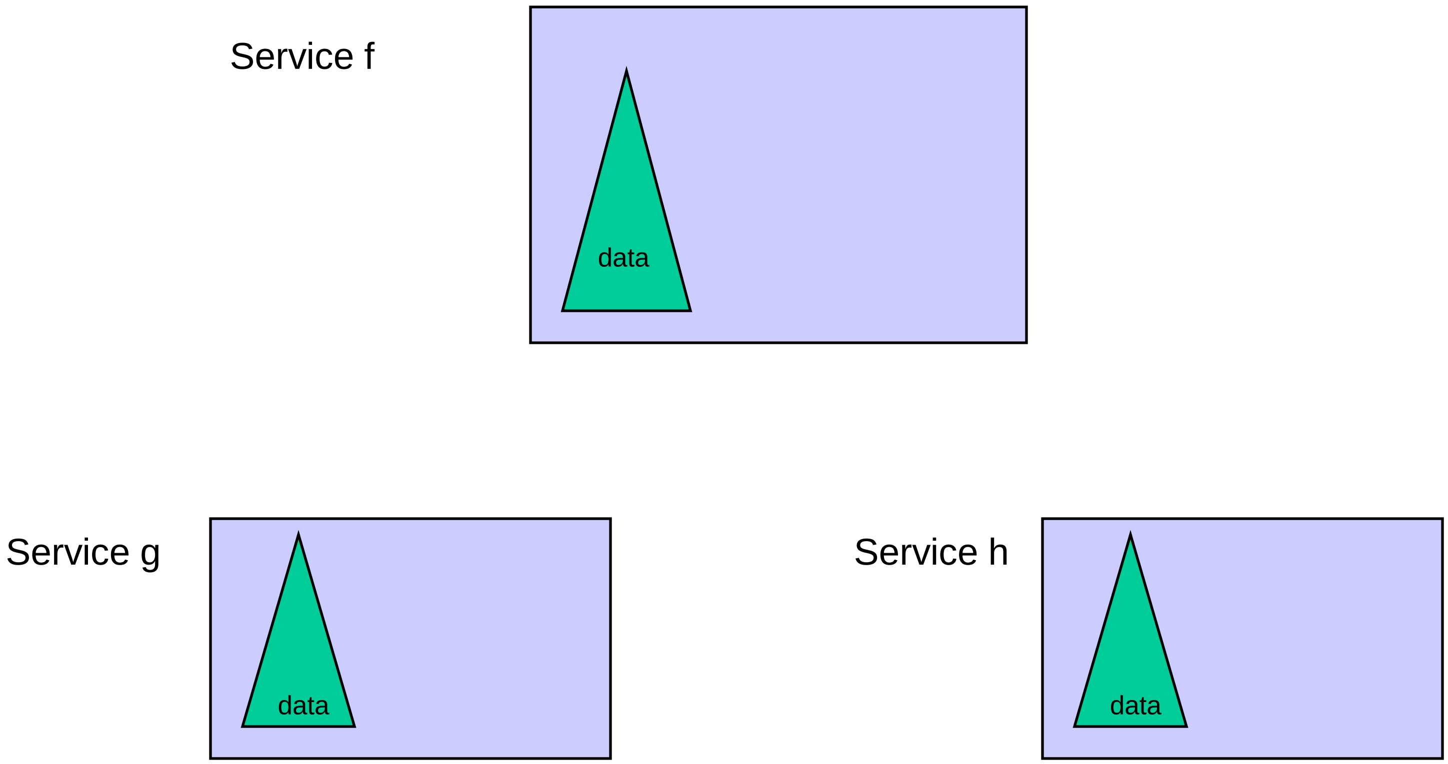
  DTDs and data constraints

- function definitions

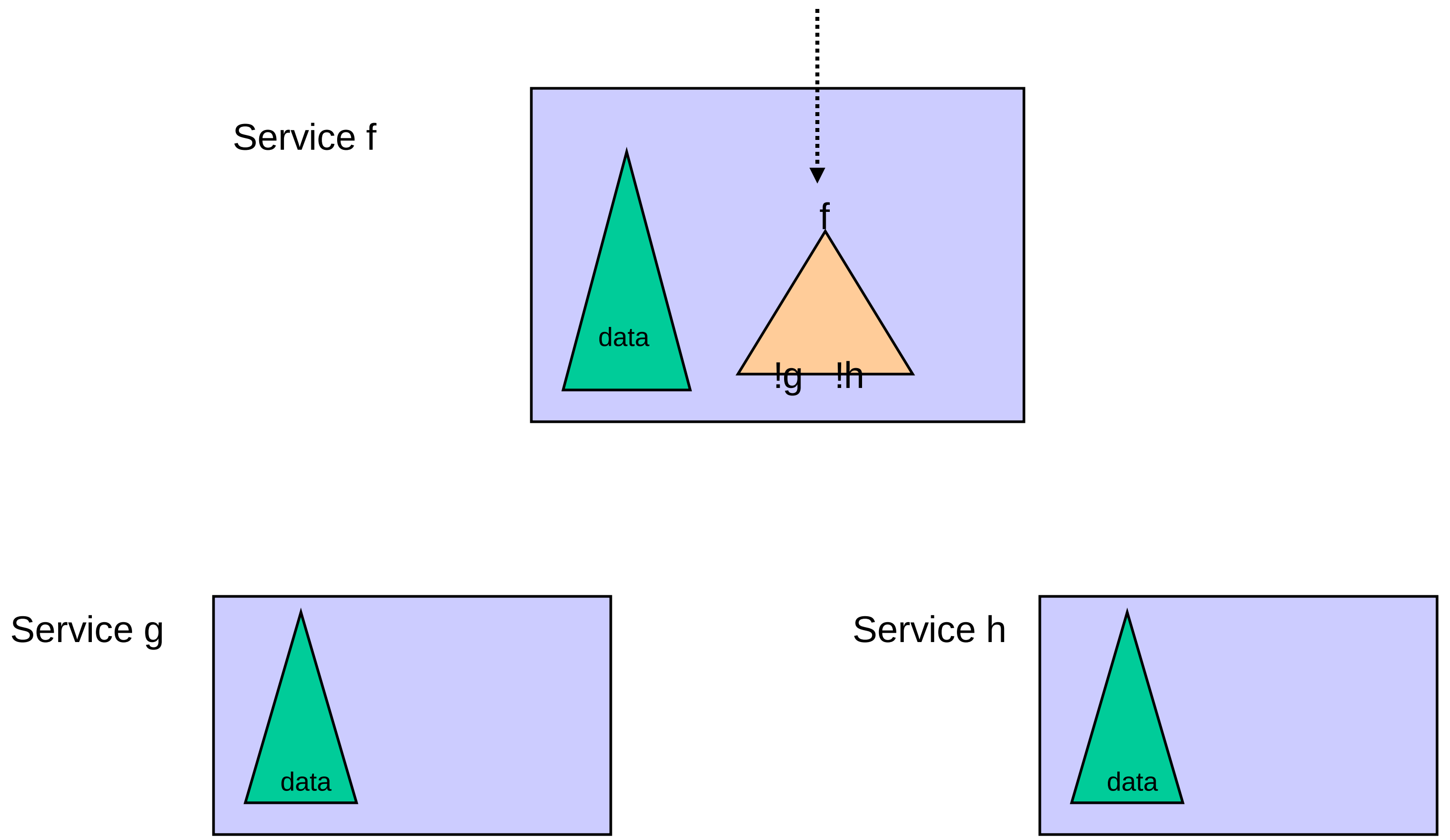
  called functions: call guards, argument queries

  supported functions: return guards, return queries

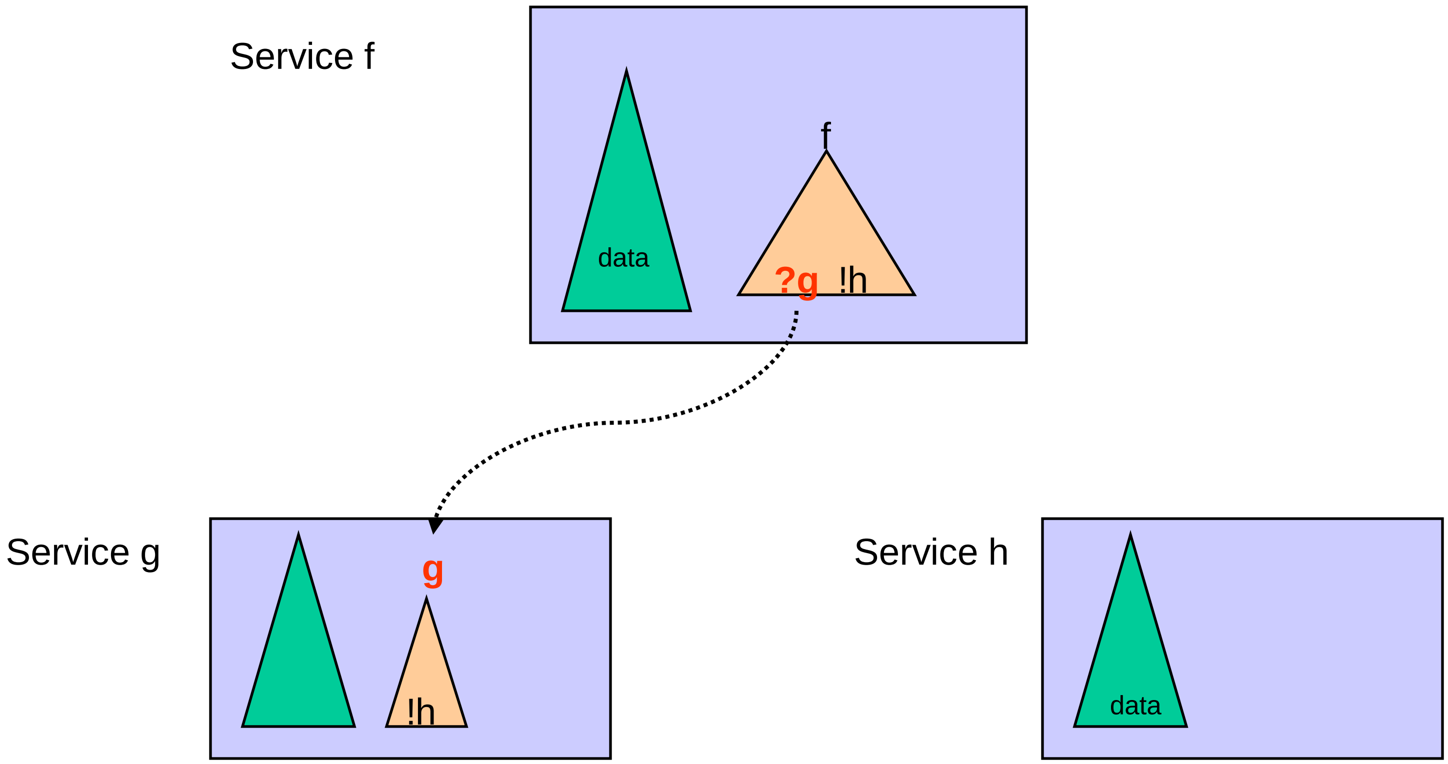
Run of a GAXML system



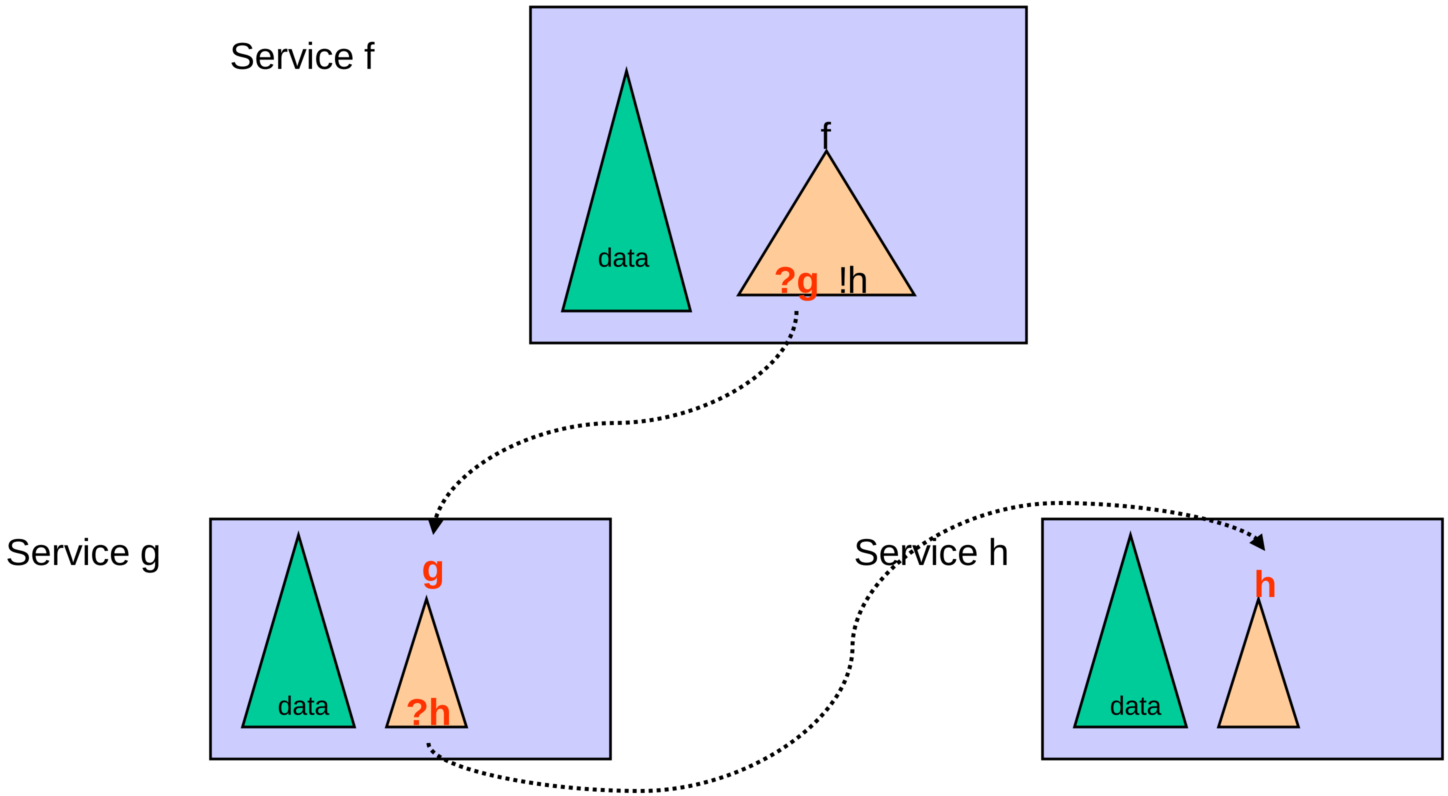
Run of a GAXML system



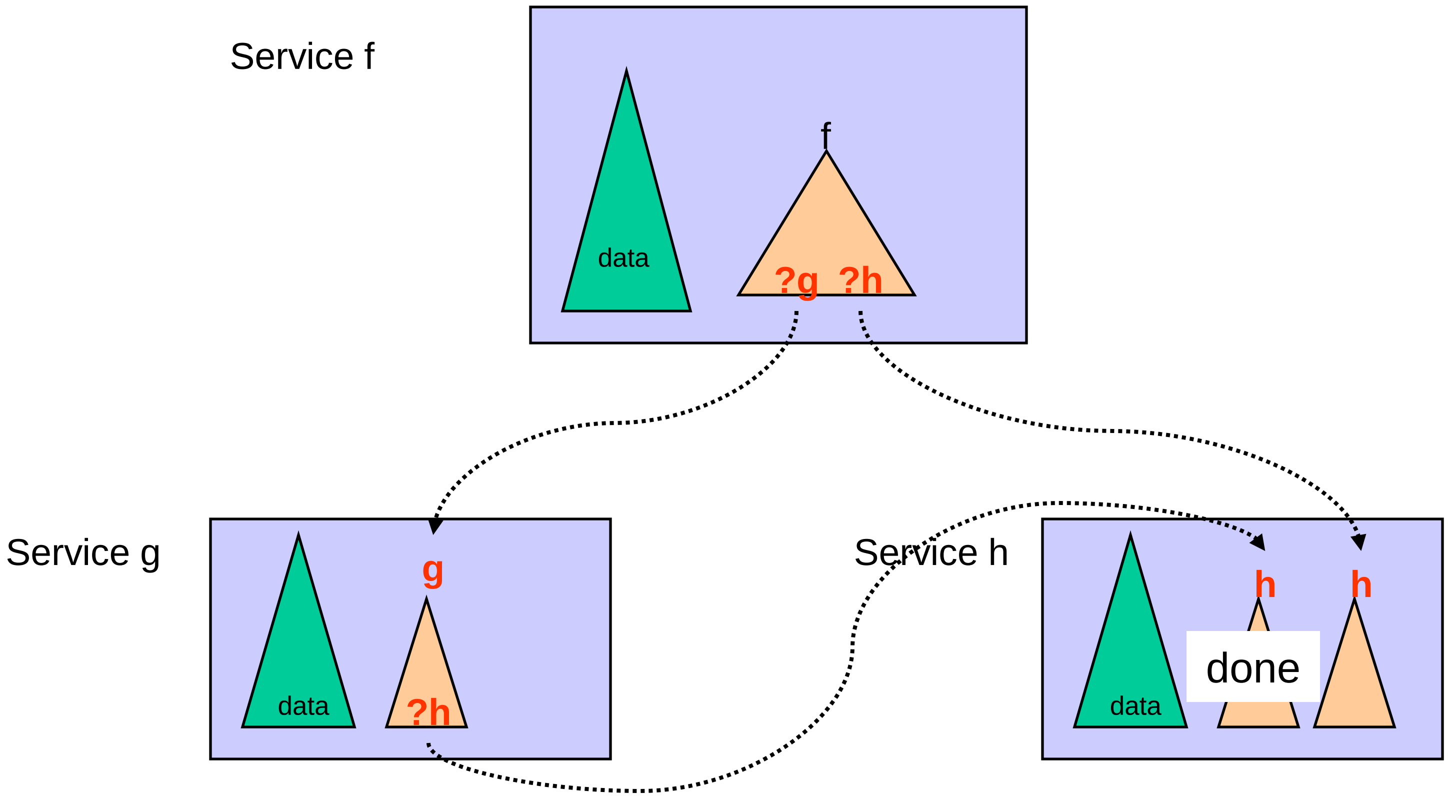
Run of a GAXML system



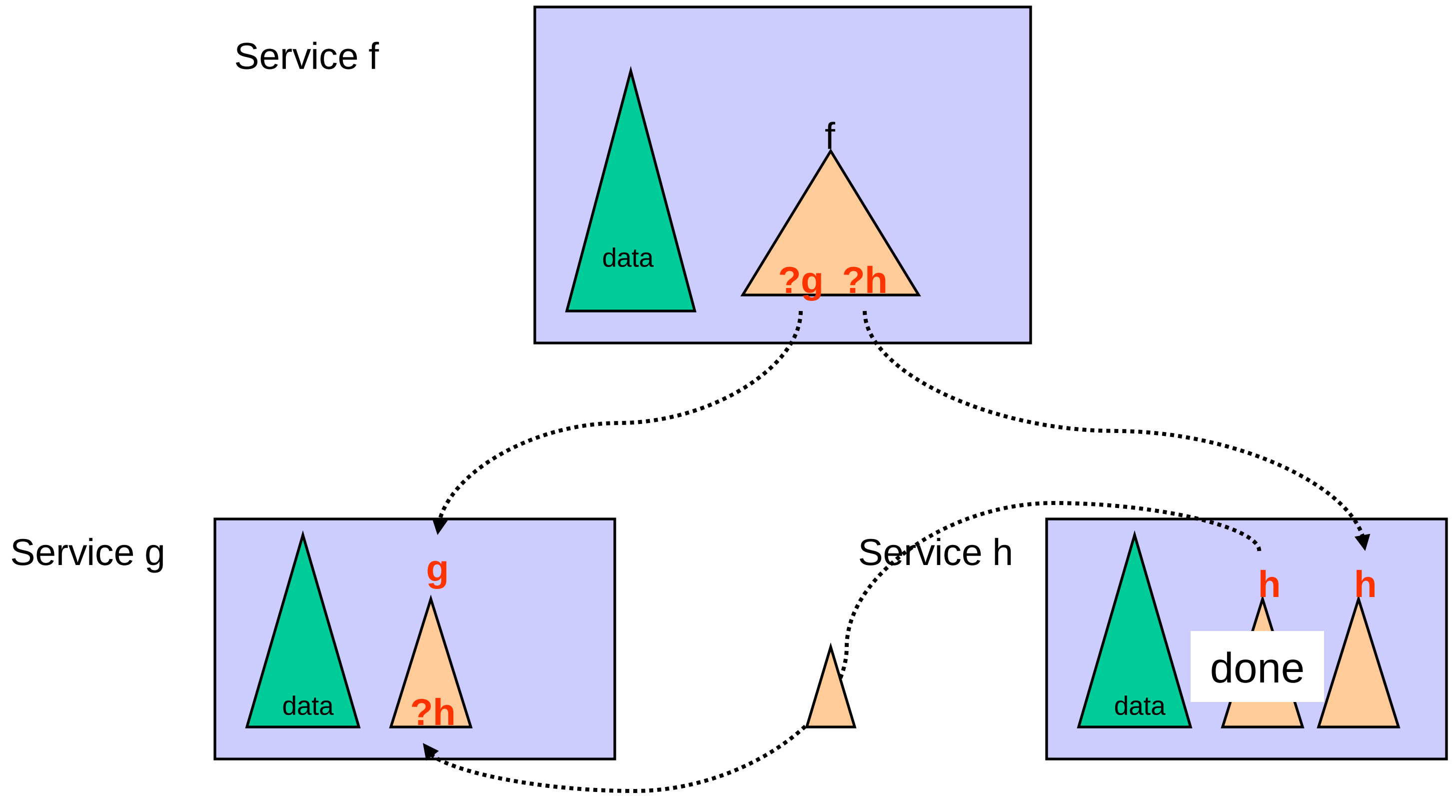
Run of a GAXML system



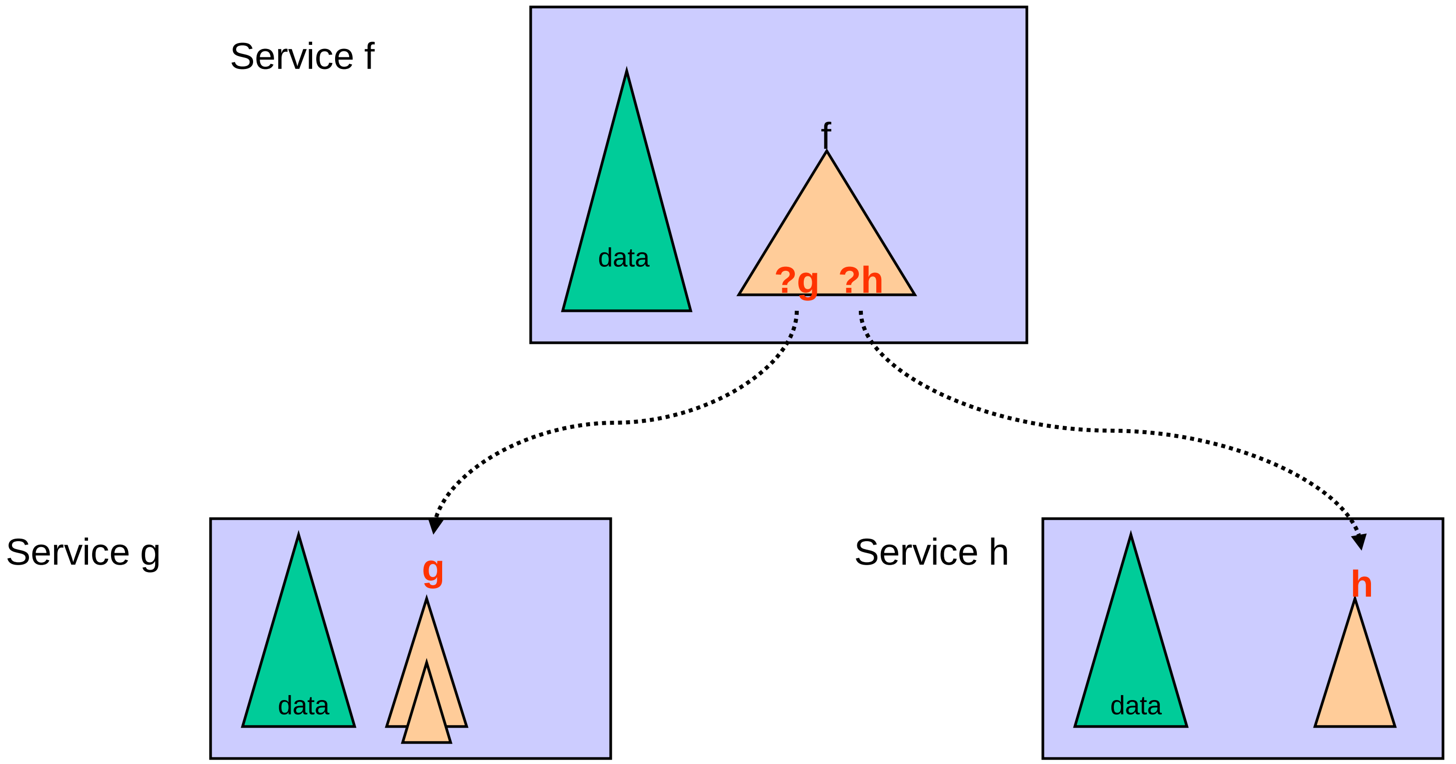
Run of a GAXML system



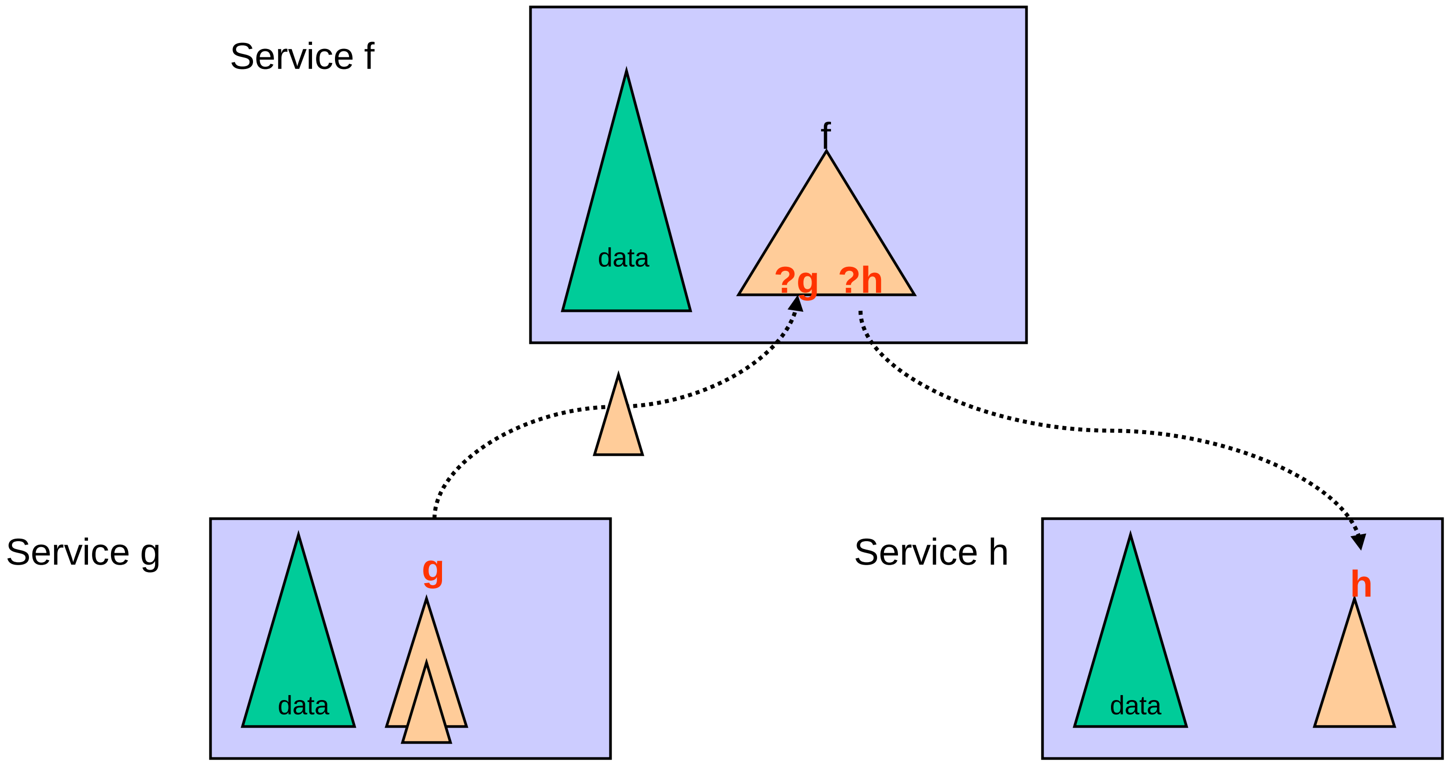
Run of a GAXML system



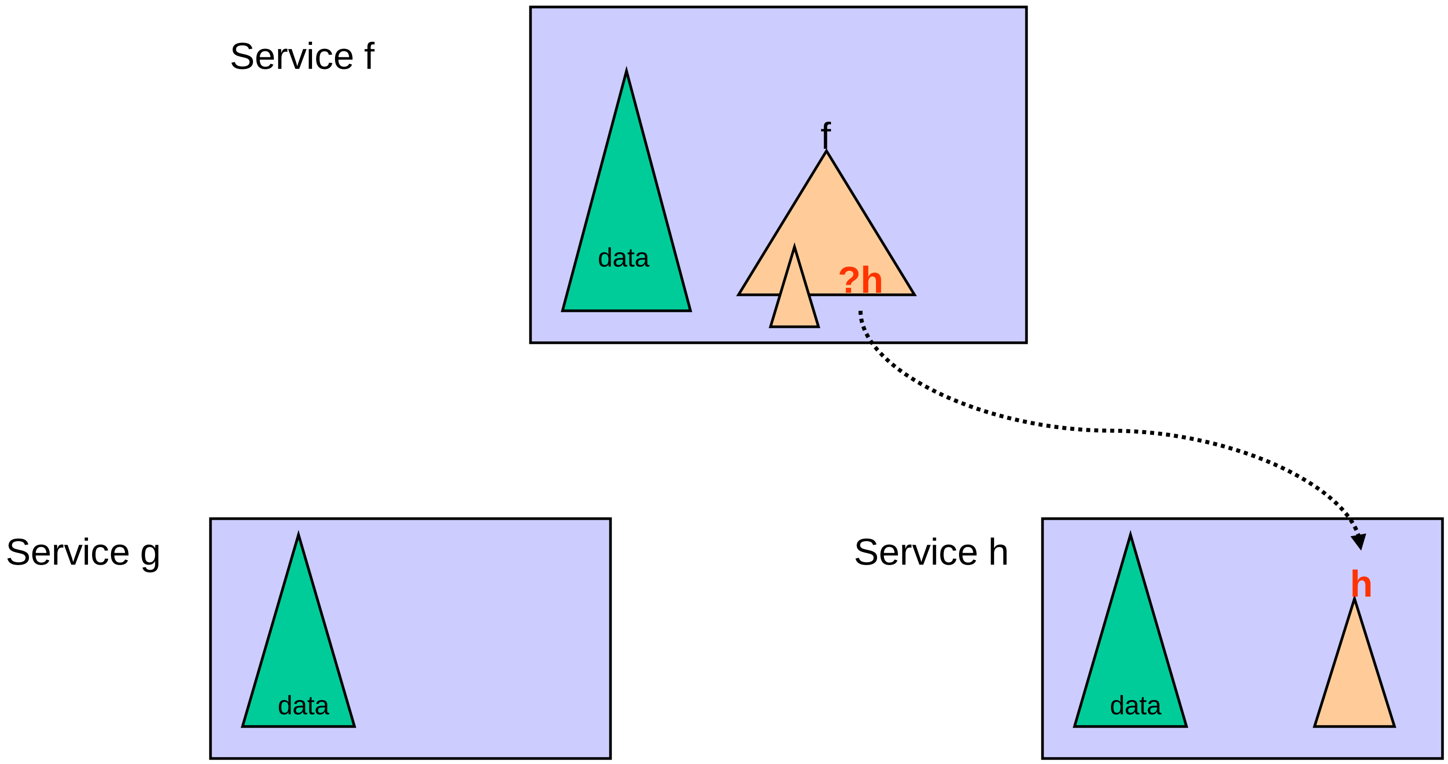
Run of a GAXML system



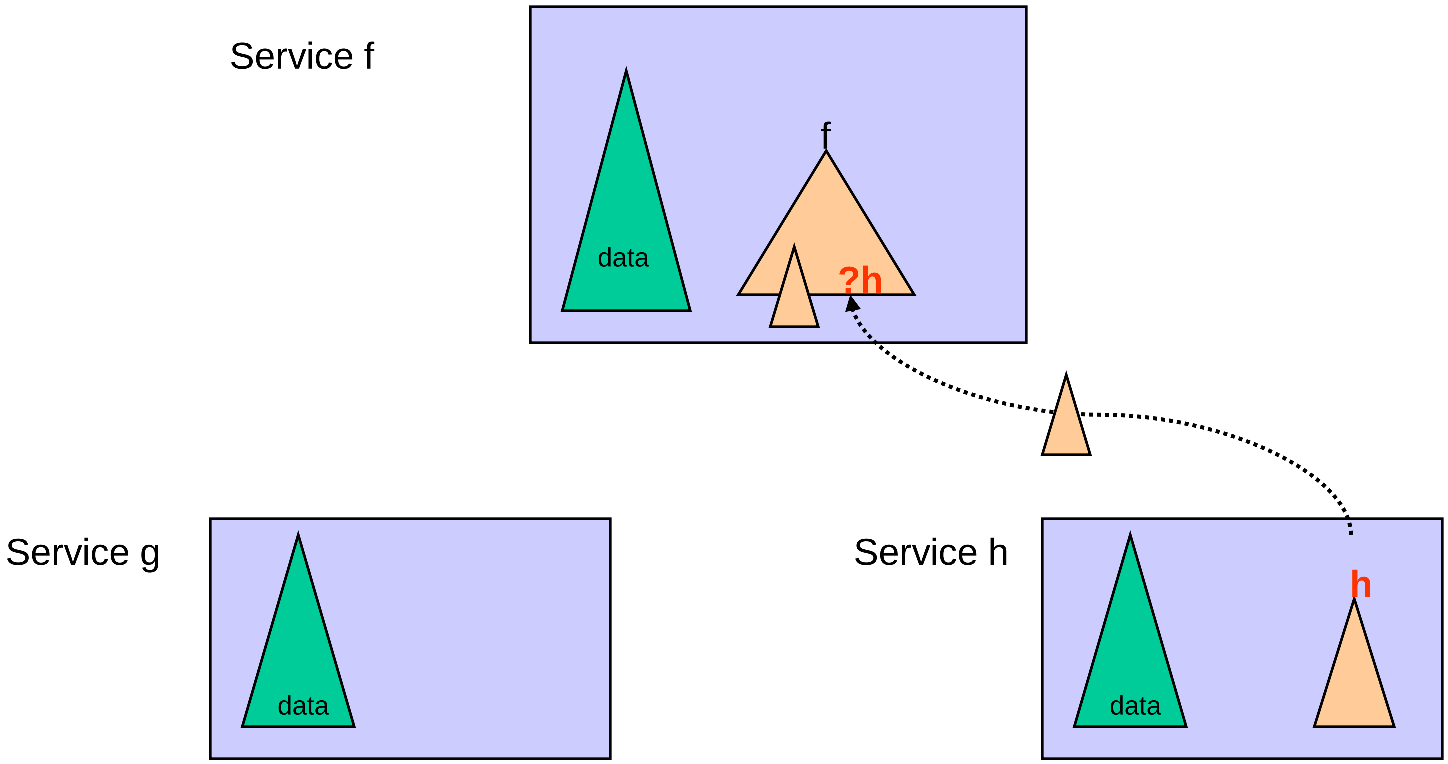
Run of a GAXML system



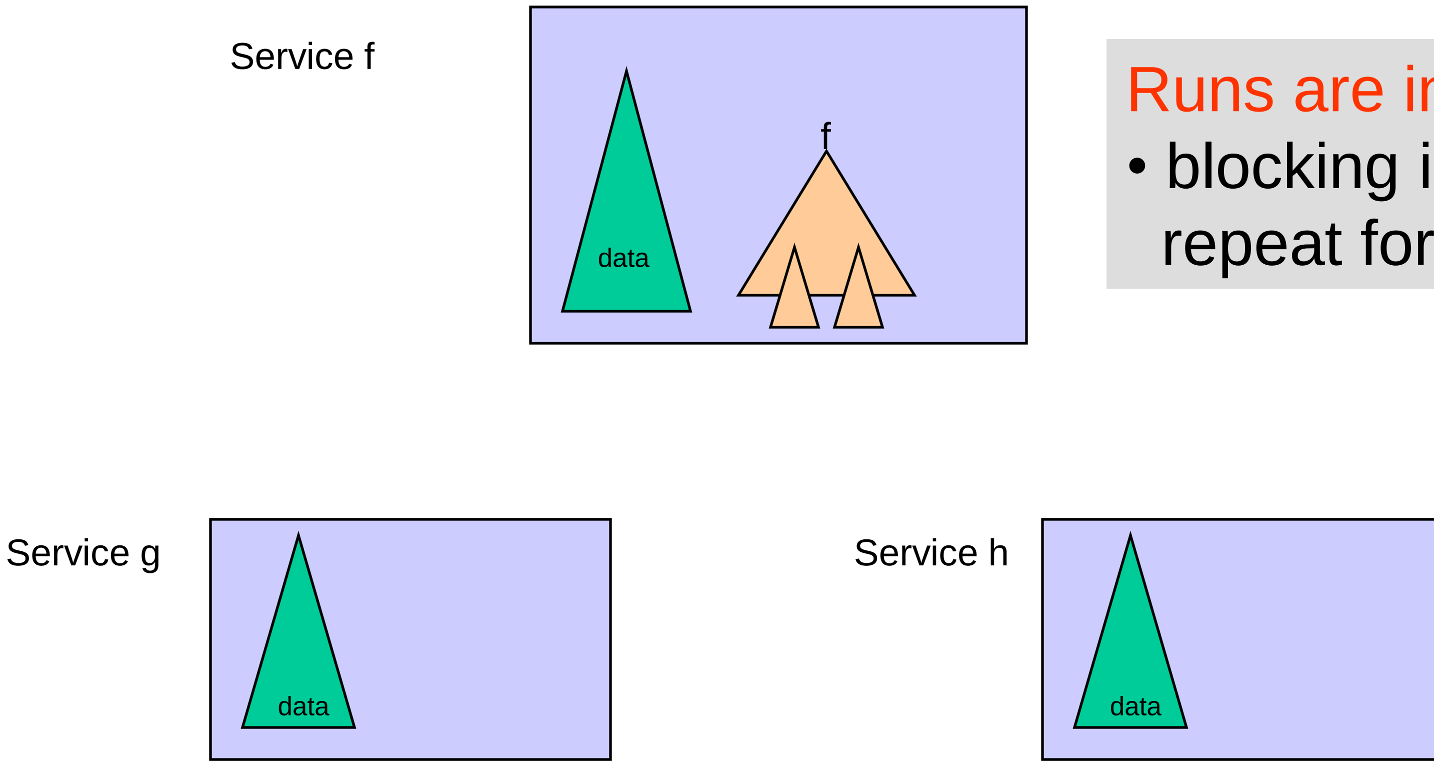
Run of a GAXML system



Run of a GAXML system



Run of a GAXML system



Runs are infinite

- blocking instances repeat forever

Specifying properties of runs: Tree-LTL

Linear-time temporal logic (LTL)

- propositions p,q,r, ...
- logical connectives: $\wedge$, $\vee$, $\neg$
- temporal operators:

G: always

F : eventually

U: until

X: next

Examples:

G p

G (p → **F** q)

GFp → **GF**q

From LTL to Tree-LTL

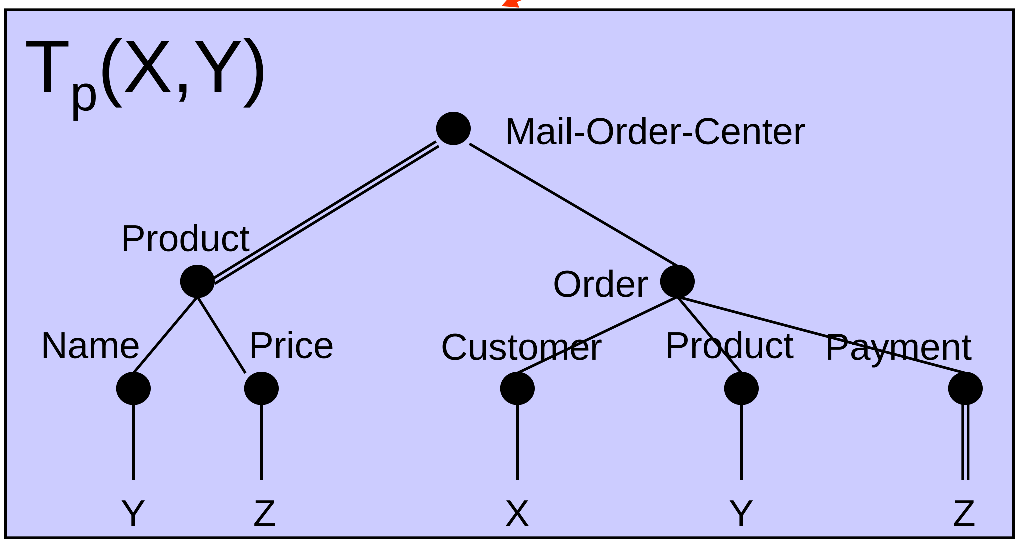
$\forall \overline{X} \; \forall \overline{Y}$ $\quad$ **G**(p $\rightarrow$ **F** q)

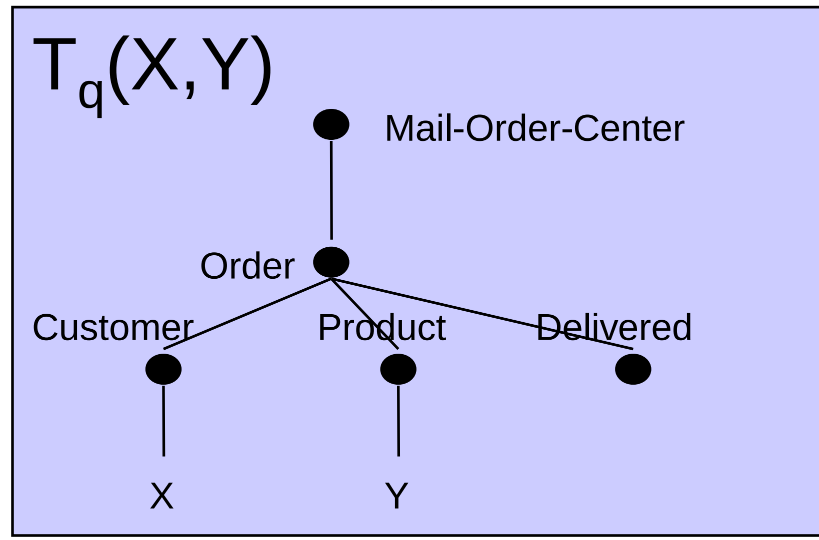
$T_p(\overline{X})$ $\quad$ $T_q(\overline{Y})$

tree patterns with free variables $\overline{X}$, $\overline{Y}$

From LTL to Tree-LTL

$\forall X \forall Y$ **G**(p → **F**q)





If a customer pays a product in the correct amount, the product is eventually delivered

Other examples of properties expressible in Tree-LTL

- no product is delivered unless it has been previously paid for in the correct amount
- the billed amount for a product is always the catalog price
- there are no two active orders by the same customer for the same product

Verification problem

Given a GAXML system S and a Tree-LTL formula φ, decide whether every run of S satisfies φ. If not, find a counterexample run.

Restriction for decidability:

non-recursiveness

syntactic restriction ensuring that every run reaches a blocking instance in a bounded number of steps

Main result

It is decidable whether a non-recursive GAXML system S satisfies a Tree-LTL formula φ

Note: still infinite-state system because of unbounded data! Use a “small run” property.

Complexity: CO-2NEXPTIME complete

- likely to be lower in many practical cases

-- CO-NEXPTIME if function call graph is a tree rather than a DAG

-- CO-NP with fixed bound on depth of trees, number of functions, and max number of variables in tree patterns

Other decidable problems for non-recursive GAXML systems

- successful termination

  does every run of S reach a blocking instance with no running function calls?

- typechecking

  if the initial instance of a run is valid, then all instances reachable in the run are also valid

valid: satisfy DTD and data constraints

Decidability with recursion

- Sufficient condition for safety with respect to Boolean combination of tree patterns φ

1. Every valid initial instance of G satisfies φ
2. If I satisfies φ and I → J, then J satisfies φ

complexity: CO-NEXPTIME

Decidability with recursion

- Bounded reachability of φ

For fixed k, is there an instance J of S that satisfies φ and is reachable by a run of length at most k ?

complexity: NEXPTIME

Conclusions

- Powerful framework for specifying evolving documents
- Tree-LTL can specify a wide range of useful properties
- Verification is decidable under the strong non-recursiveness restriction
- However, non-recursiveness is common in practice
- Even for recursive GAXML systems, one can isolate and verify meaningful non-recursive fragments

  Example:

  individual orders in the order processing system

Current and future work

- Extensions of verification results

  allow some recursion
- Connection to other workflow specification mechanisms
- Use AXML as a model for

  business artifacts

  influential IBM proposal for data-centric workflows

CO-2NEXPTIME upper bound

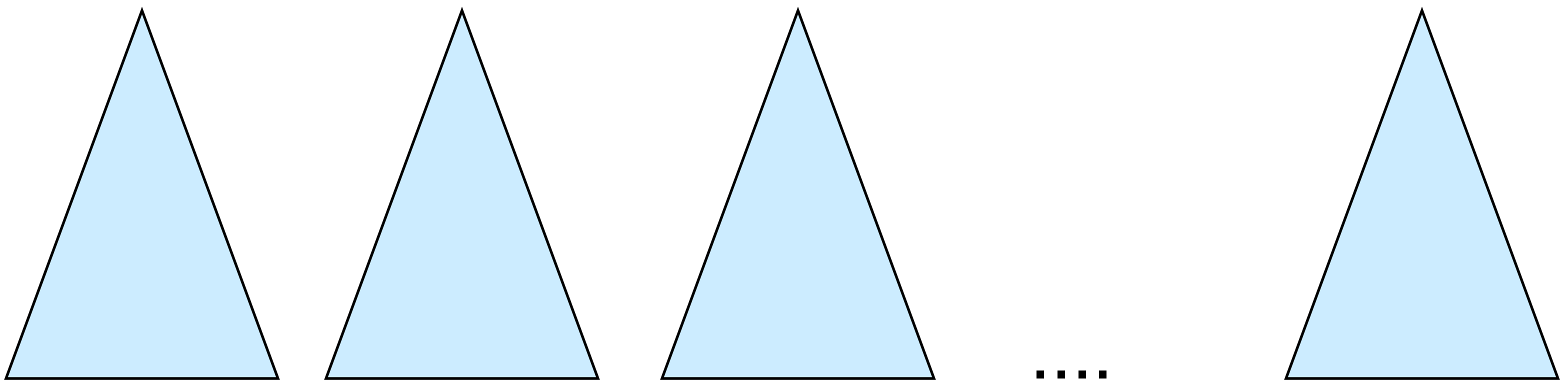
- Main idea: prove a "small run" property

If there is a run of S violating φ, then there is a "small run" of S violating φ

- size of "small run": exponential length, with instances doubly exponential in S and φ

Proof: reminiscent of small model property for $\exists^*\forall^*$ FO sentences

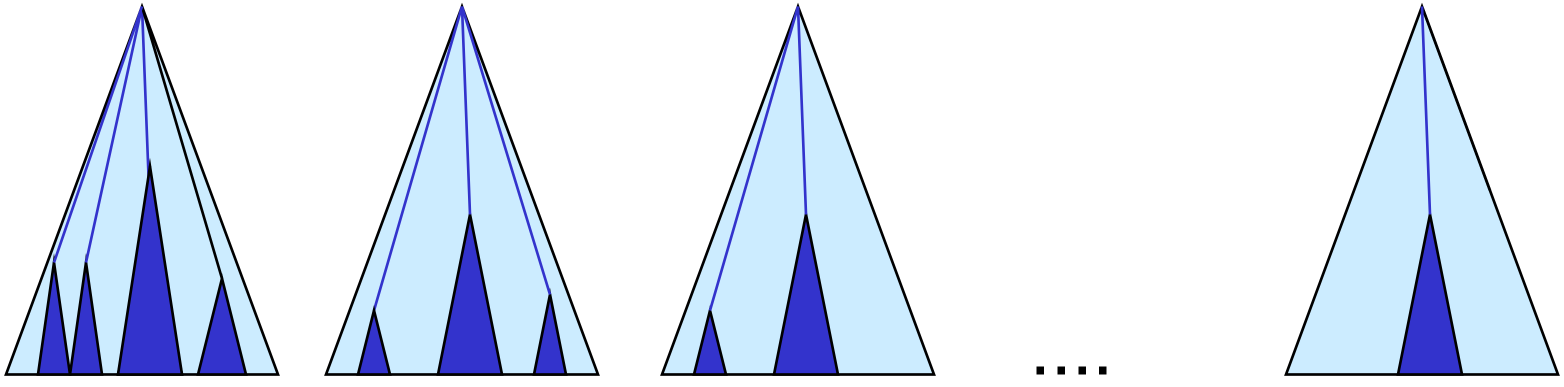
model built from witnesses to $\exists^*$ quantifiers



blocking run satisfying $\neg \varphi$

Proof: reminiscent of small model property for $\exists^*\forall^*$ FO sentences

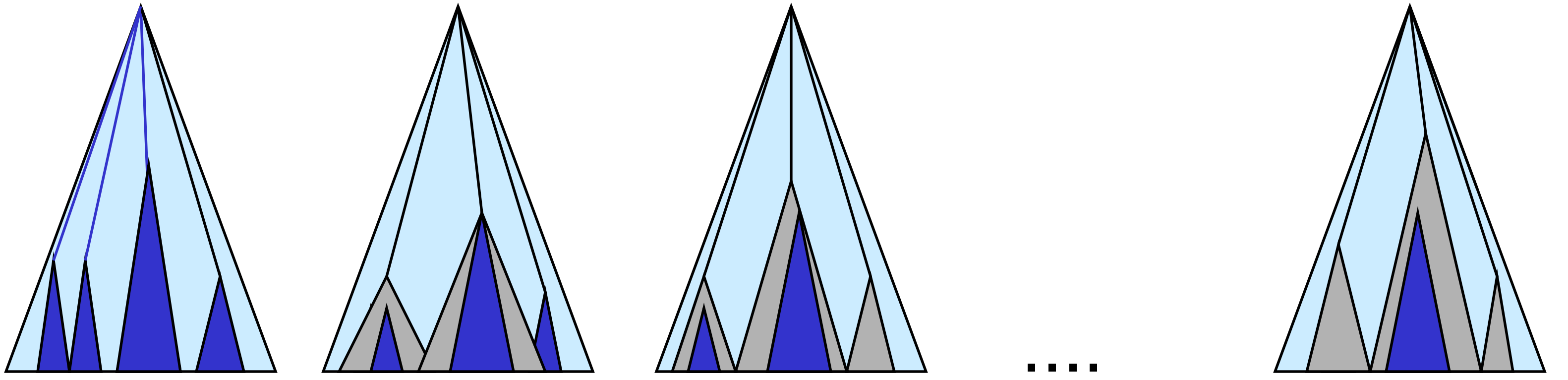
model built from witnesses to $\exists^*$ quantifiers



small run built in two stages: first collect witnesses needed to enable transitions and satisfy $\neg\varphi$

Proof: reminiscent of small model property for $\exists^*\forall^*$ FO sentences

model built from witnesses to $\exists^*$ quantifiers



then construct real run from witnesses

CO-2NEXPTIME lower bound

- Simulation of 2NEXPTIME Turing machine

2NEXPTIME Turing machine M and word w

↓

GAXML system S and Tree-LTL formula φ

S violates φ iff M accepts w

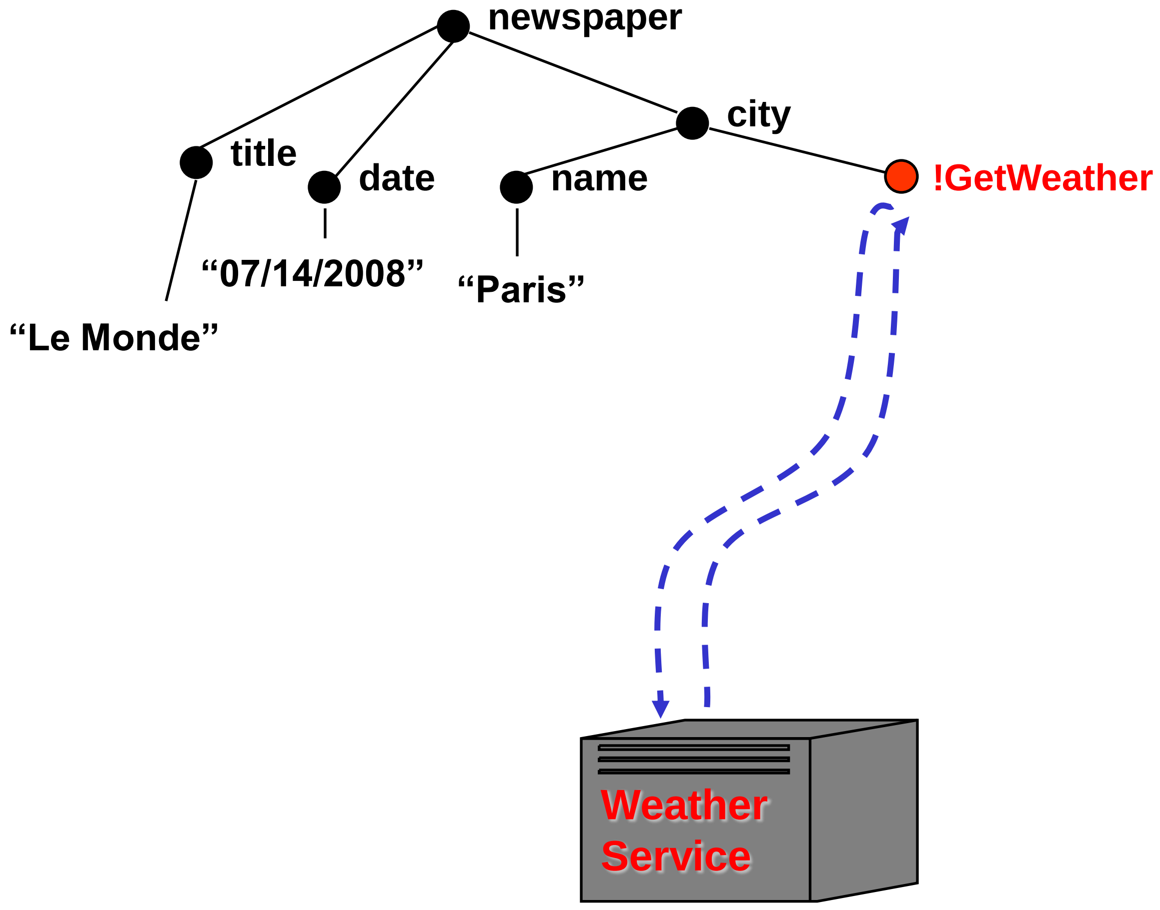
Non-trivial simulation:
tricky encoding and control needed

Non-recursive GAXML system

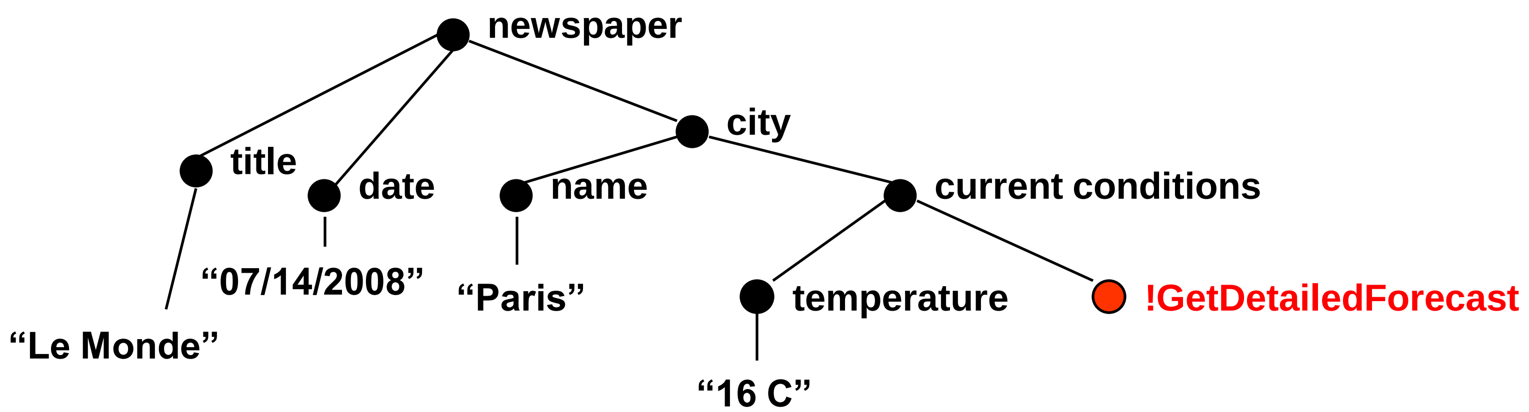
- acyclic function call graph
- no continuous functions
- non-recursive DTD
- DTD allows bounded number of function calls in each valid tree

Note: still infinite-state system!

Keeping dynamic data fresh



Keeping dynamic data fresh



Example: argument query for **!Bill**