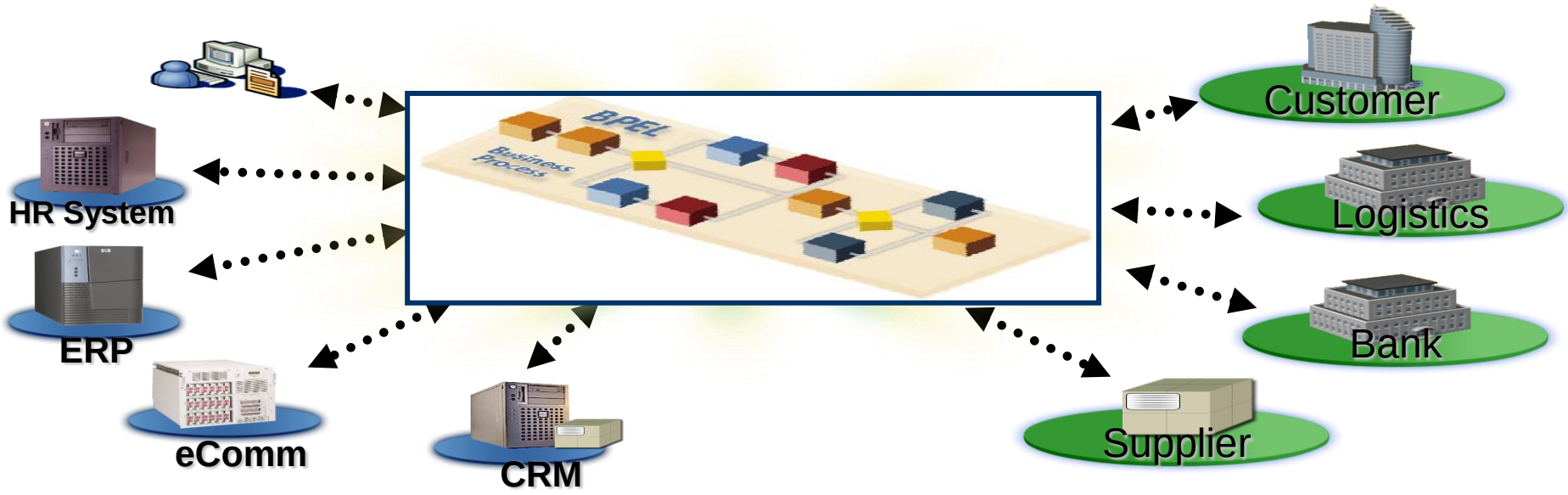
Querying Past and Future in Web Applications

Daniel Deutch Tova Milo

Tel-Aviv University



Outline

- Introduction & Motivation
- Querying Future

  [VLDB'06, ICDE'09, ICDT'09, VLDB'09]
- Querying Past
- **[VLDB'07, VLDB'08, ICDT'09, VLDB'09]**
- Related and Future work

Outline

- Introduction & Motivation
- Querying Future

  [VLDB'06, ICDE'09, ICDT'09, VLDB'09]
- Querying Past
- **[VLDB'07, VLDB'08, ICDT'09, VLDB'09]**
- Related and Future work

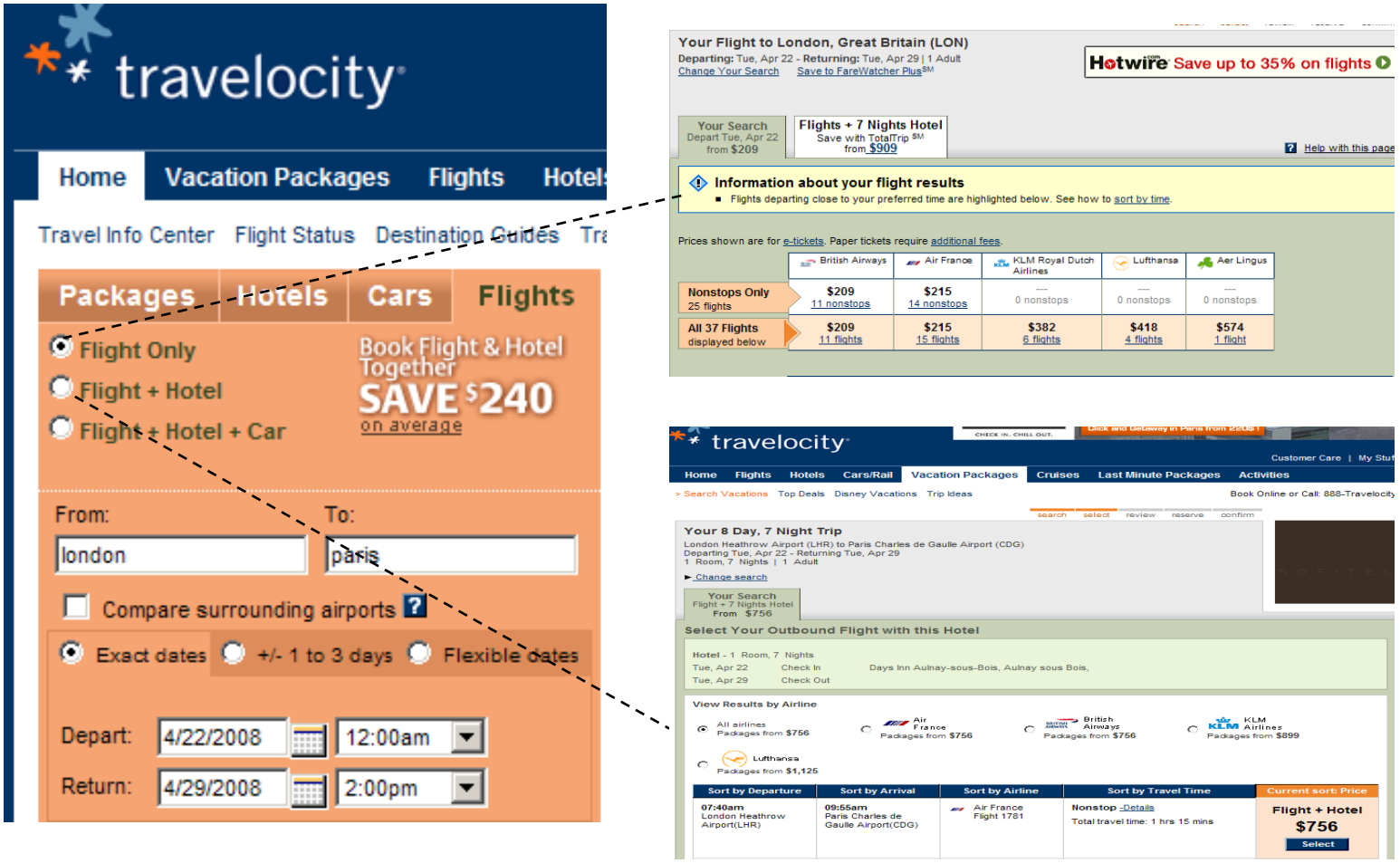
Querying Web applications

- Web-based Business Processes (BPs) are very popular
- Querying the past
  - past executions of a given application

    for improving business logic, optimization, personalized ads,…
  - past design patterns of applications

    to exploit when building a new application
- Querying (possible) future executions
  - the above + verification

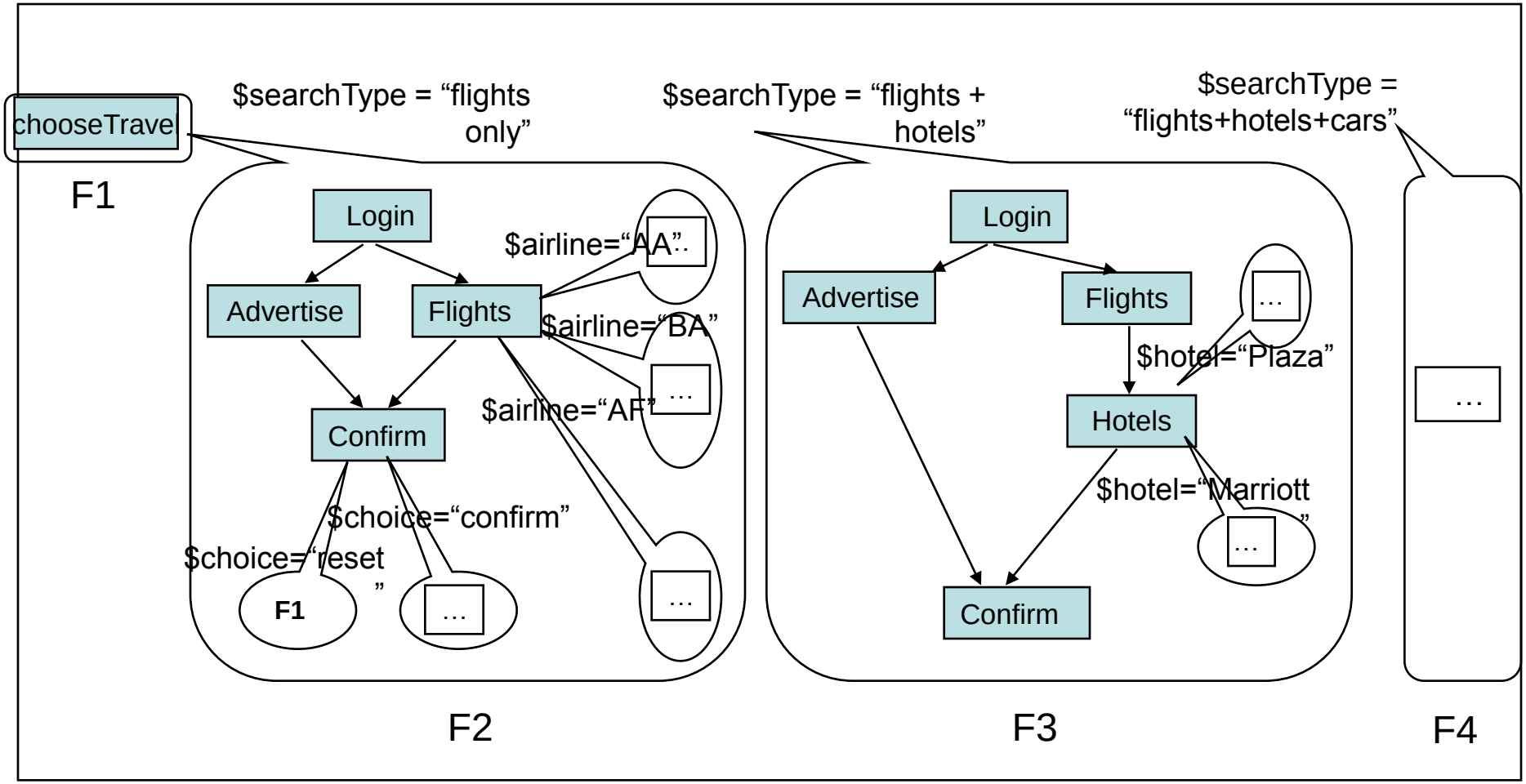
Example Web Application



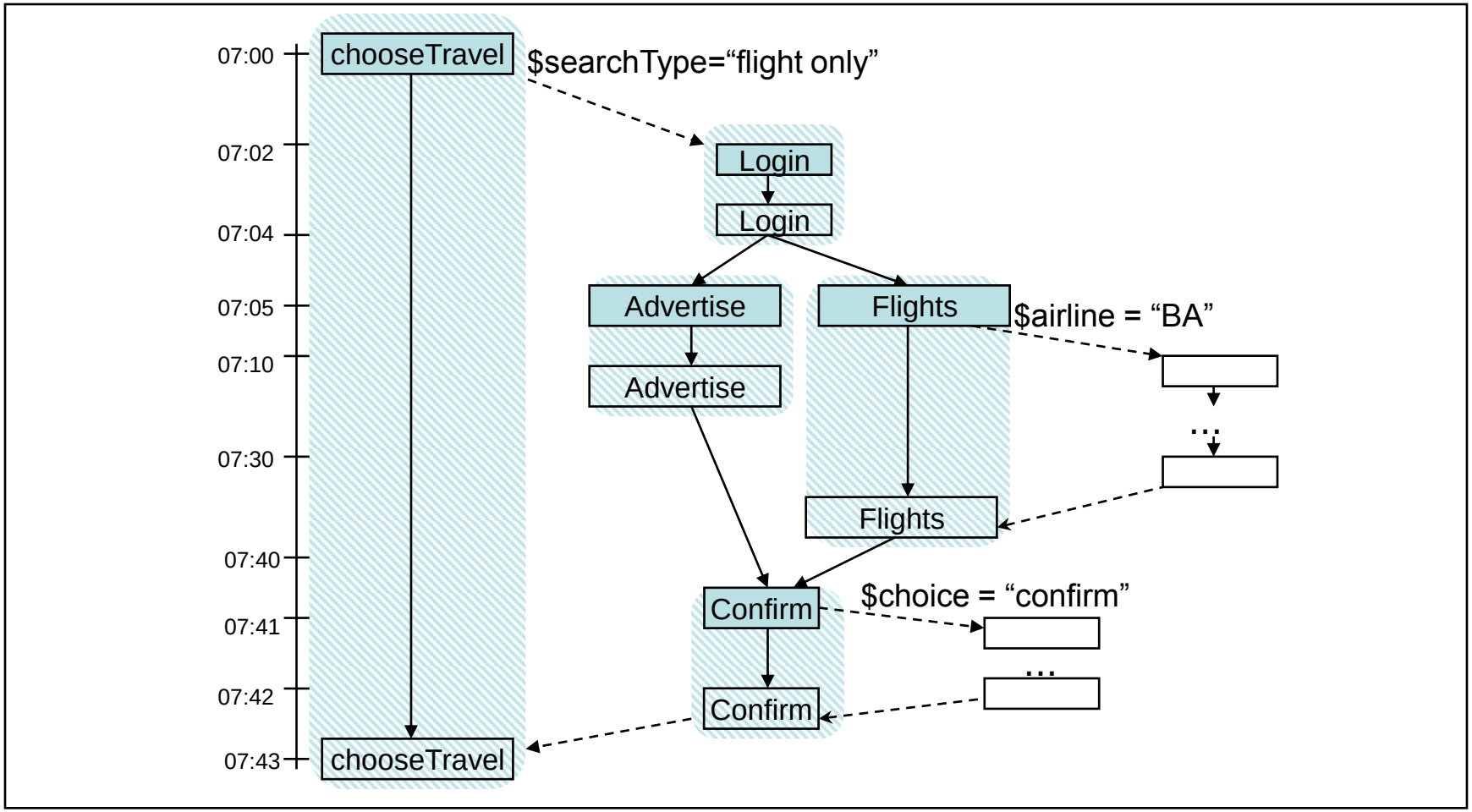
Application Specification

- BP specifications (e.g. in **BPEL**) are compiled to running code
  => Queries over specification structure
- Modeled as nested DAGs
  Each DAG corresponds to a function\web-page
- **Nodes** model activities (activation and completion)
  Edges mark flow relation
  Atomic/compound activities
  Nesting models implementation relations \ links
  Guarding formulas (on external events) model choices
- Recursion is allowed

Example Specification



Example Execution Flow



Sources of Uncertainty

- Partial Tracing, due to lack of storage, confidentiality, etc.
- External Effects, e.g. user choices, server response time
  - The effect of external events is modeled by logical formulas, guarding implementations
  - At run-time, formulas' truth values determines the chosen implementations

Thus the **past** is uncertain, and the **future** unknown

Outline

- Introduction & Motivation
- Querying Future
- Querying Past
- Related and Future work

Querying Future

Queries on possible EX-flows

- Can a user reserve a flight without giving her credit card details?
- What is the probability that this would happen?
- How can this happen / what are the most common scenarios?
- What is the common behavior of users that search for an Air France flight+hotel deal but quit without making a reservation?

  Focus on the search-related sub-flow.
- What is the best way to get a cheap flight+hotel deal, or one that maximizes the FF millage/points ?

Query types

- Boolean (+ probability)
- Selection (+ top-k ranking)
- Projection (+ top-k ranking)

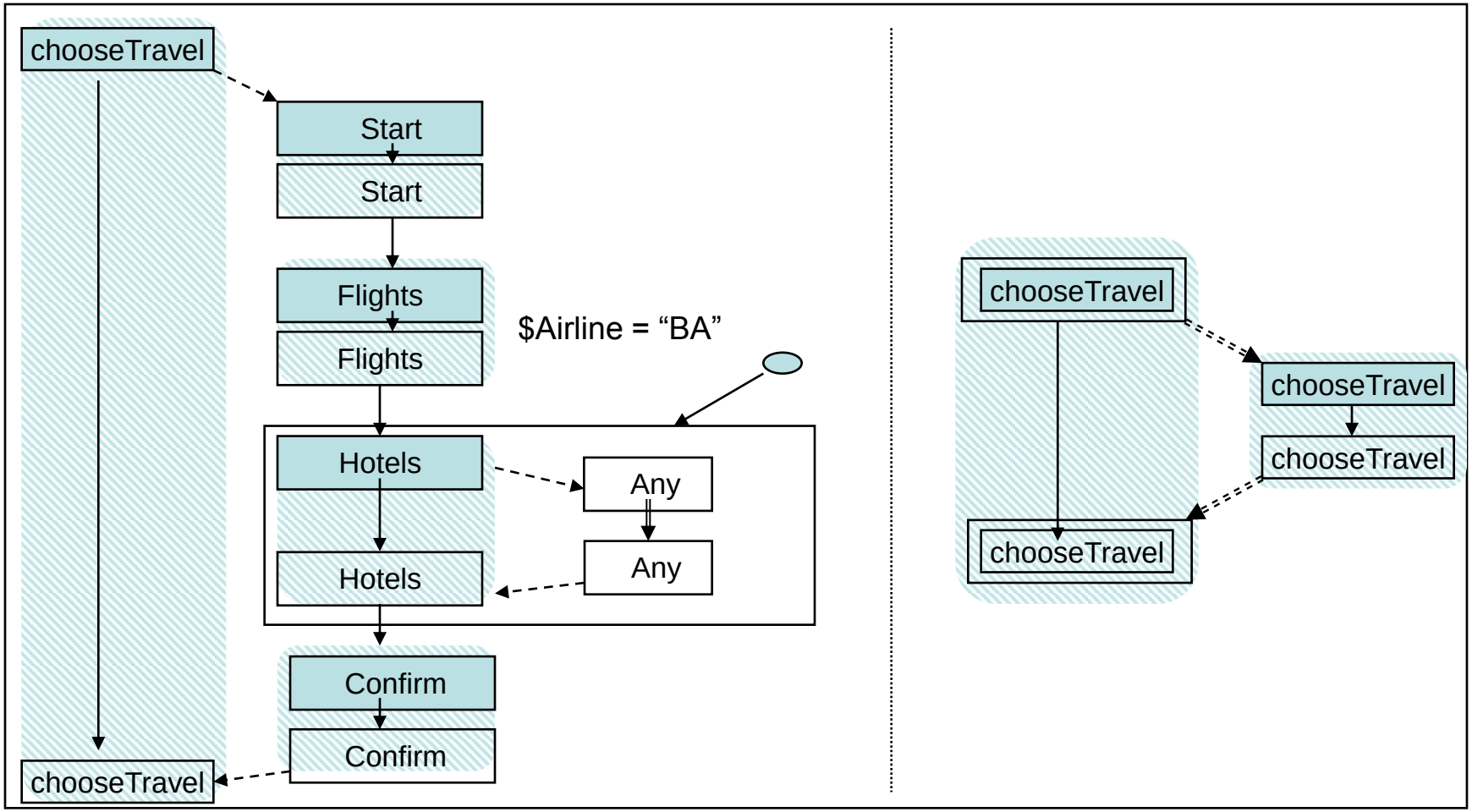
Query types

- Boolean (+ probability) easy (hard)
- Selection (+ top-k ranking) easy (easy)
- Projection (+ top-k ranking) easy (hard)

Query language

- Execution patterns
- Intuitive, similar in structure to execution flows
- Seek for occurrences (homomorphism) of the pattern within execution flow DAGs
- May contain transitive nodes and edges
- May contain a projection part

Example Query

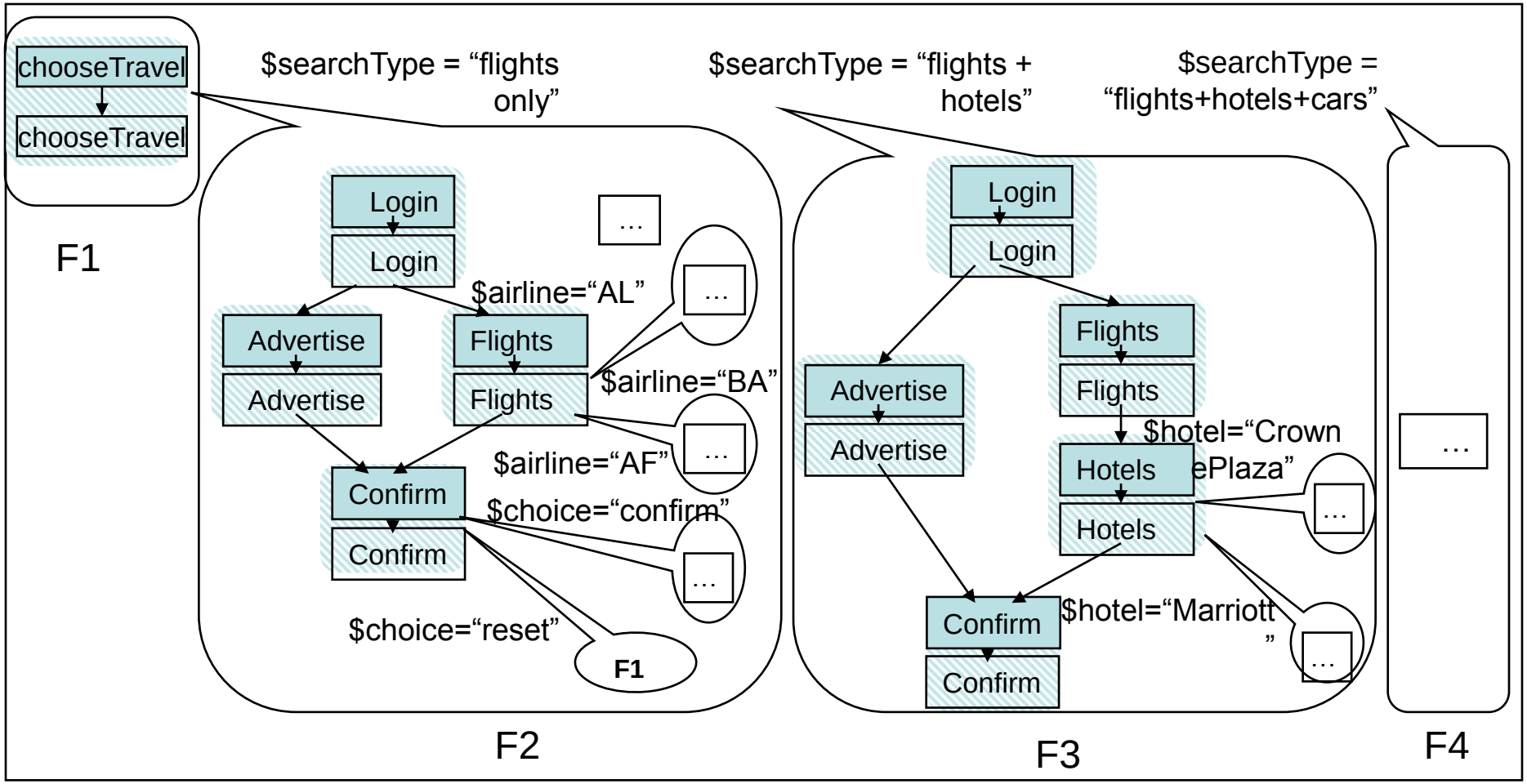


Weighting flows

- A weighted model
  - cWeight - choice Weight **(product cost, shipping time, likelihood)**
  - Aggr **(sum, multiplication)**
  - fWeight - flow Weight **(total cost, shipping time,…)**
- Varying sensitivity to flow history (cWeight)
- Varying level of monotonicity (fWeight)

Querying Future

Our Example Specification



Example (Weight = Likelihood)

- We distinct three classes of probability distributions, according to the level of dependency of cWeight.
- **History independence** (markovian):
  No dependencies between formulas.
- **Bounded-history**:
  Dependency in(at most) B last formula values.
- **Unbounded-history**

Sample of Results (top-k selection)

- Complexity of query evaluation depends on cWeight
  - PTIME in the spec., exponential in history bound and query.
  - NP hard in both.
  - Undecidable for unbounded history
- (Instance) Optimality of our algo depends on fWeight
  - Strongly monotone: Optimal
  - Semi-strongly monotone: Instance optimal
    No optimal exists
  - Weakly monotone: Not (instance) optimal
    No instance optimal exists

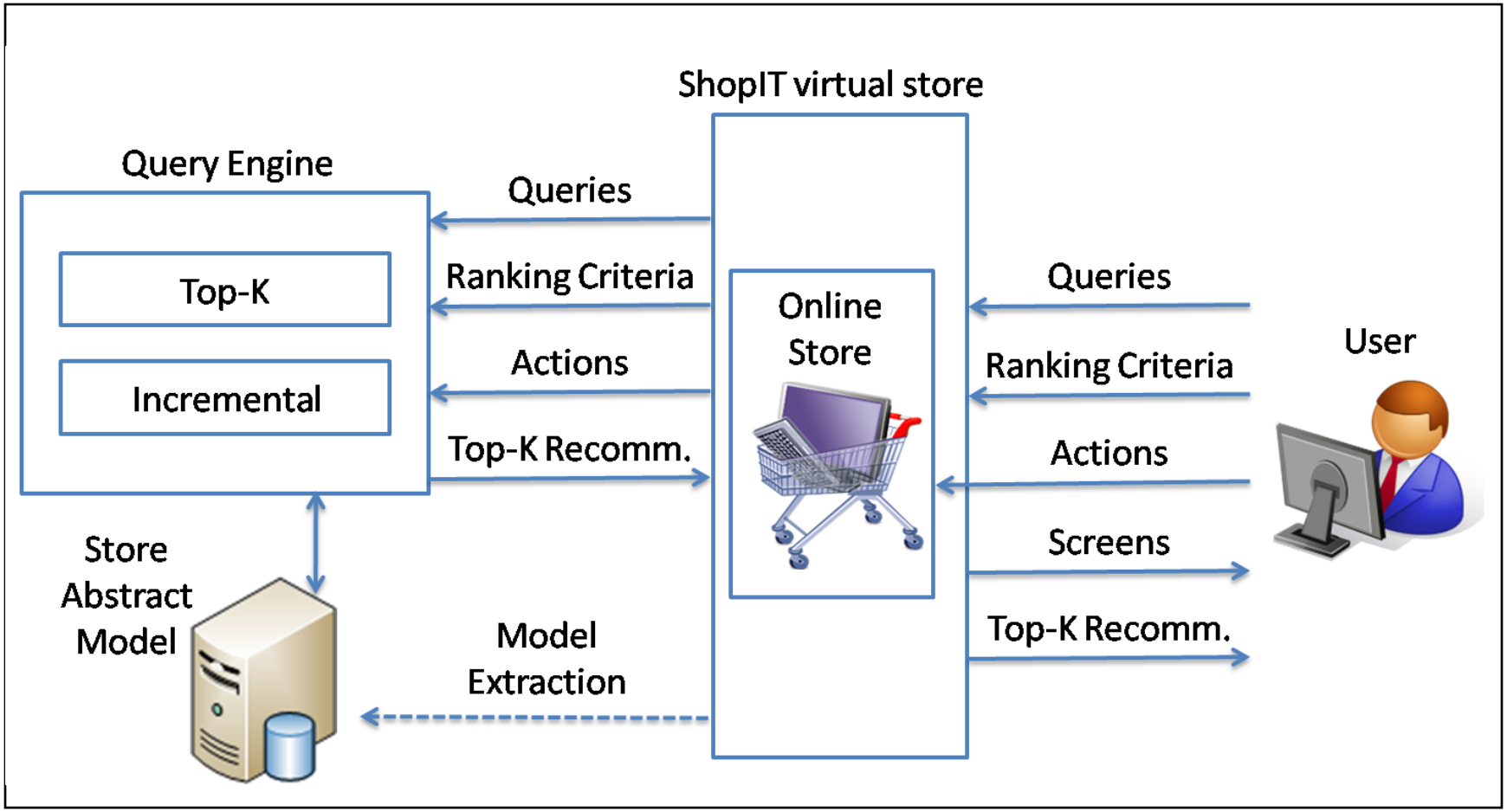
Sample of Results (Boolean)

- Harder:
  - Need to sum up probability of (a possibly infinite number of) qualifying flows
- Computing exact probability is impossible
- Approximation is possible
  - Technique: representing probabilities via set of linear equations
  - EXPTIME in general (NP Hard)
  - PTIME for non-recursive apps

Sample of Results (top-k projection)

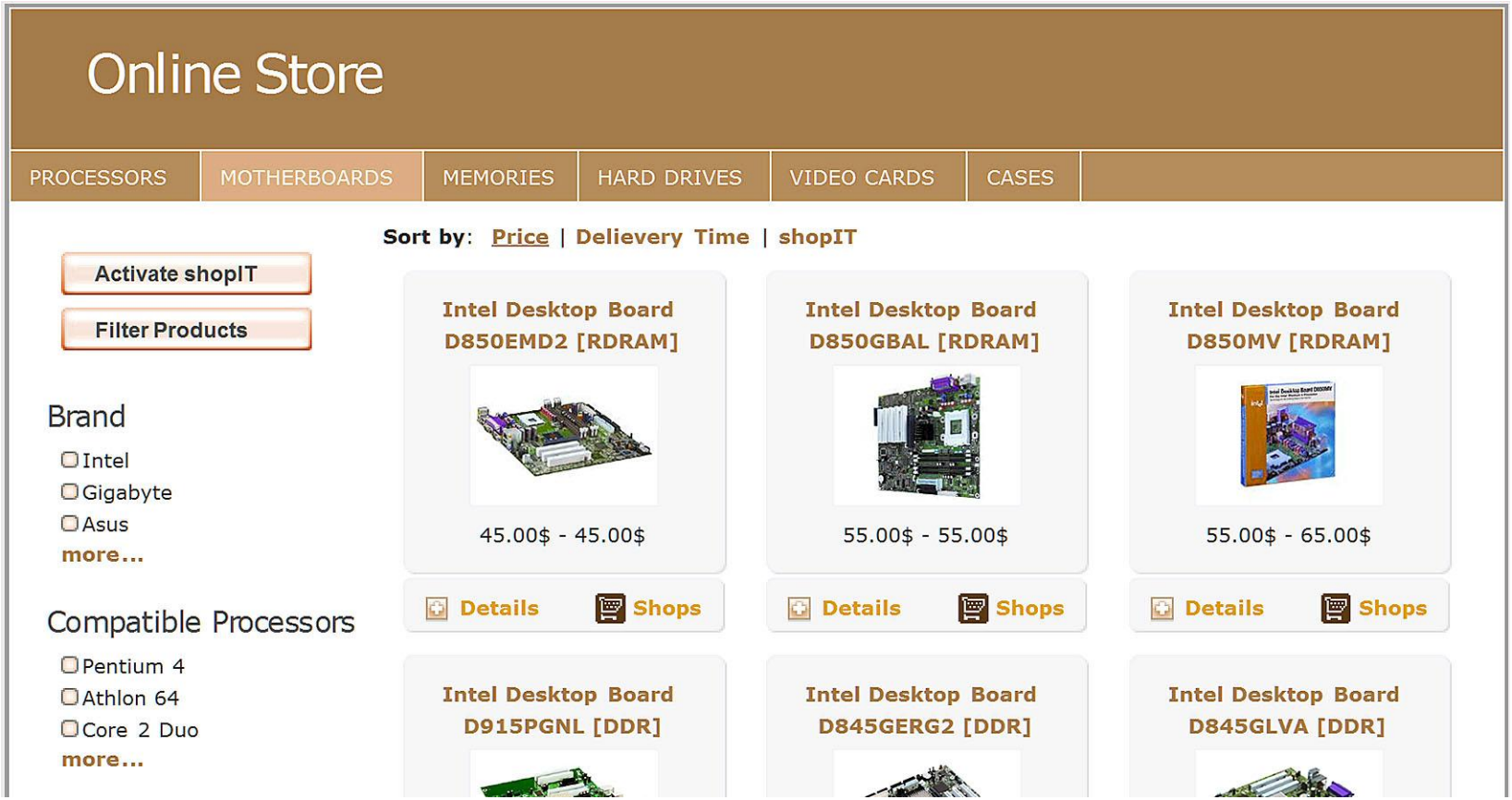
- Source of difficulty:
  - need to consider a possibly infinite # of answers,
  - and sum up probabilities of their possibly infinite # of origins
- Technique:
  - “Small world” theorem allows to consider only a bounded number of answers,
  - then use (Boolean queries) Oracle
- EXPTIME even for non recursive apps (NP Hard)

ShopIT – Shopping assistanT



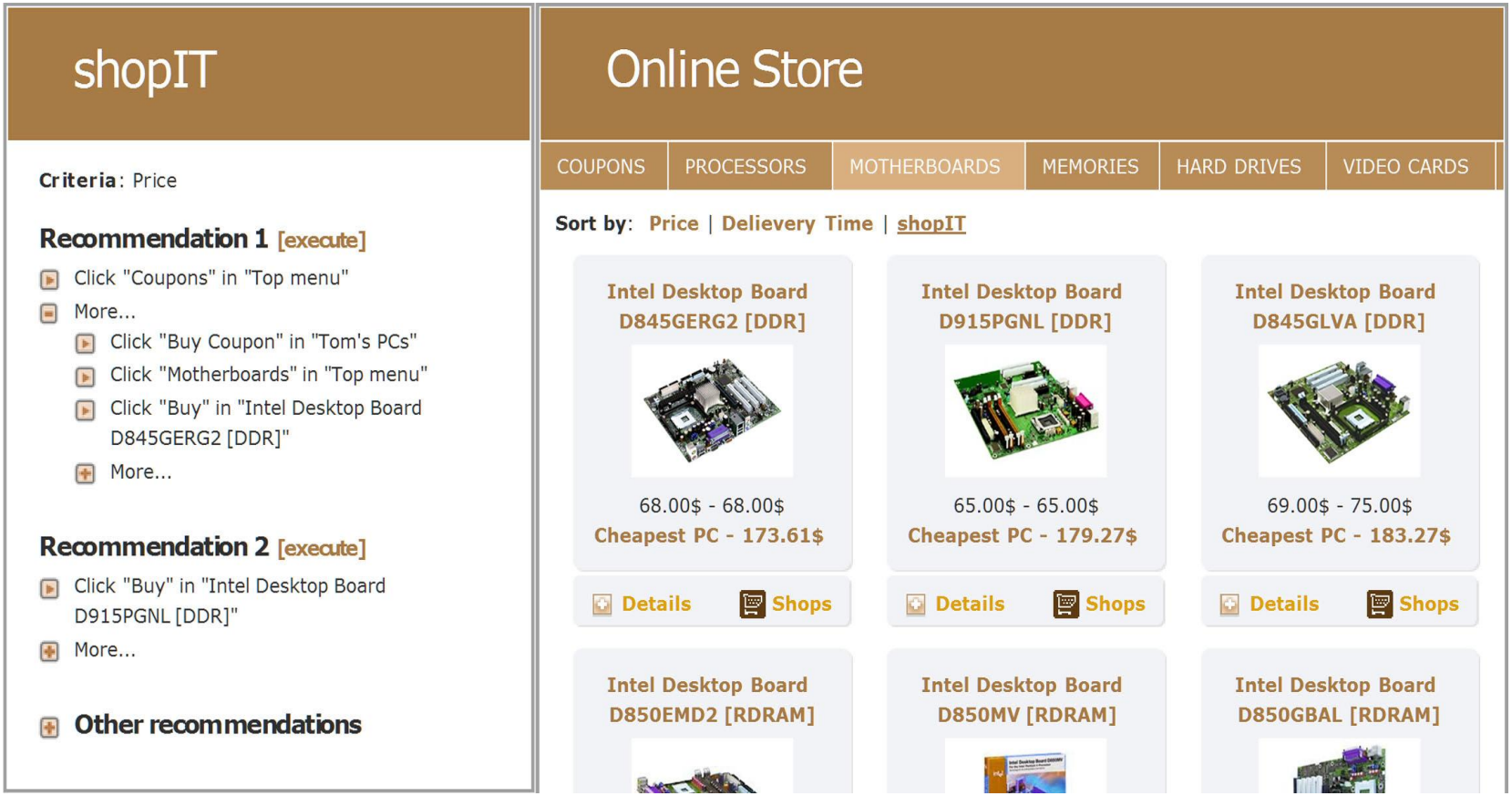
Querying Future

ShopIT – Shopping assistanT



Querying Future

ShopIT – Shopping assistanT



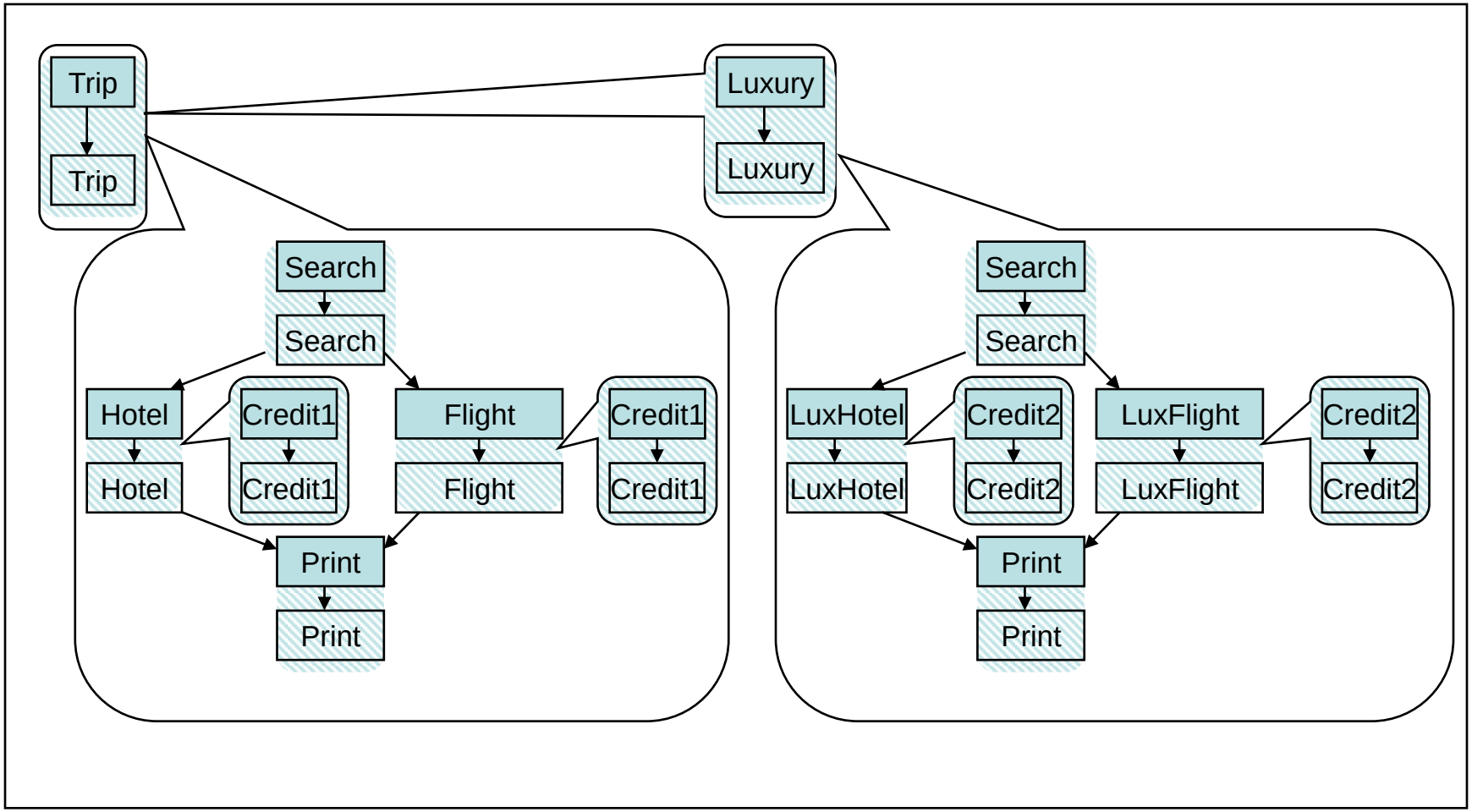
Outline

- Introduction & Motivation
- Querying Future
- Querying Past
- Related and Future work

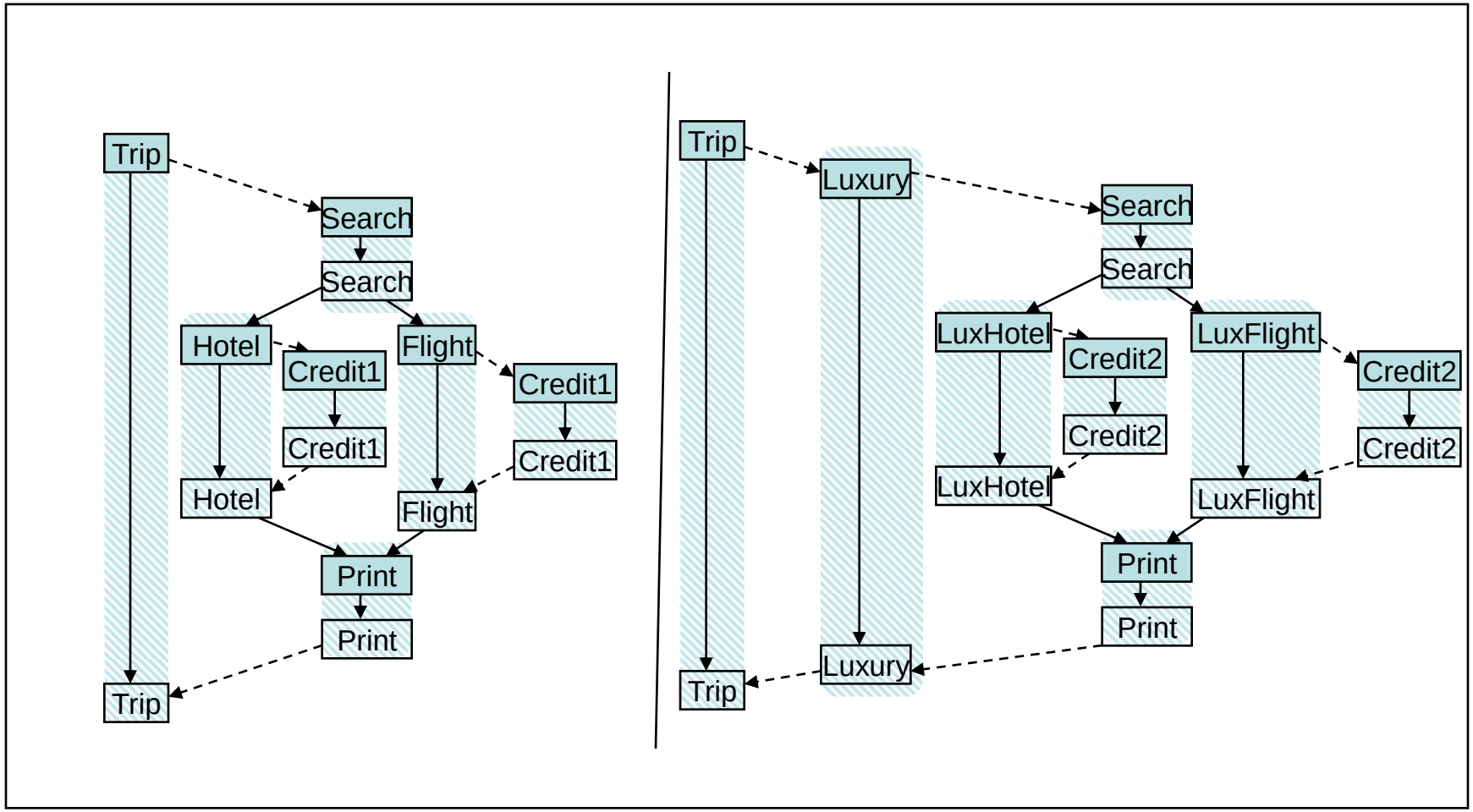
Types of Partial Traces

- Partial Tracing, due to lack of storage, confidentiality,…
  - **Naïve** tracing records all activities accurately
  - **Semi-Naïve** tracing contains only partial information on the names of some activities
  - **Selective** tracing may omit some activities occurrences
- Tracing systems (called types) are represented by a renaming function and a deletion set

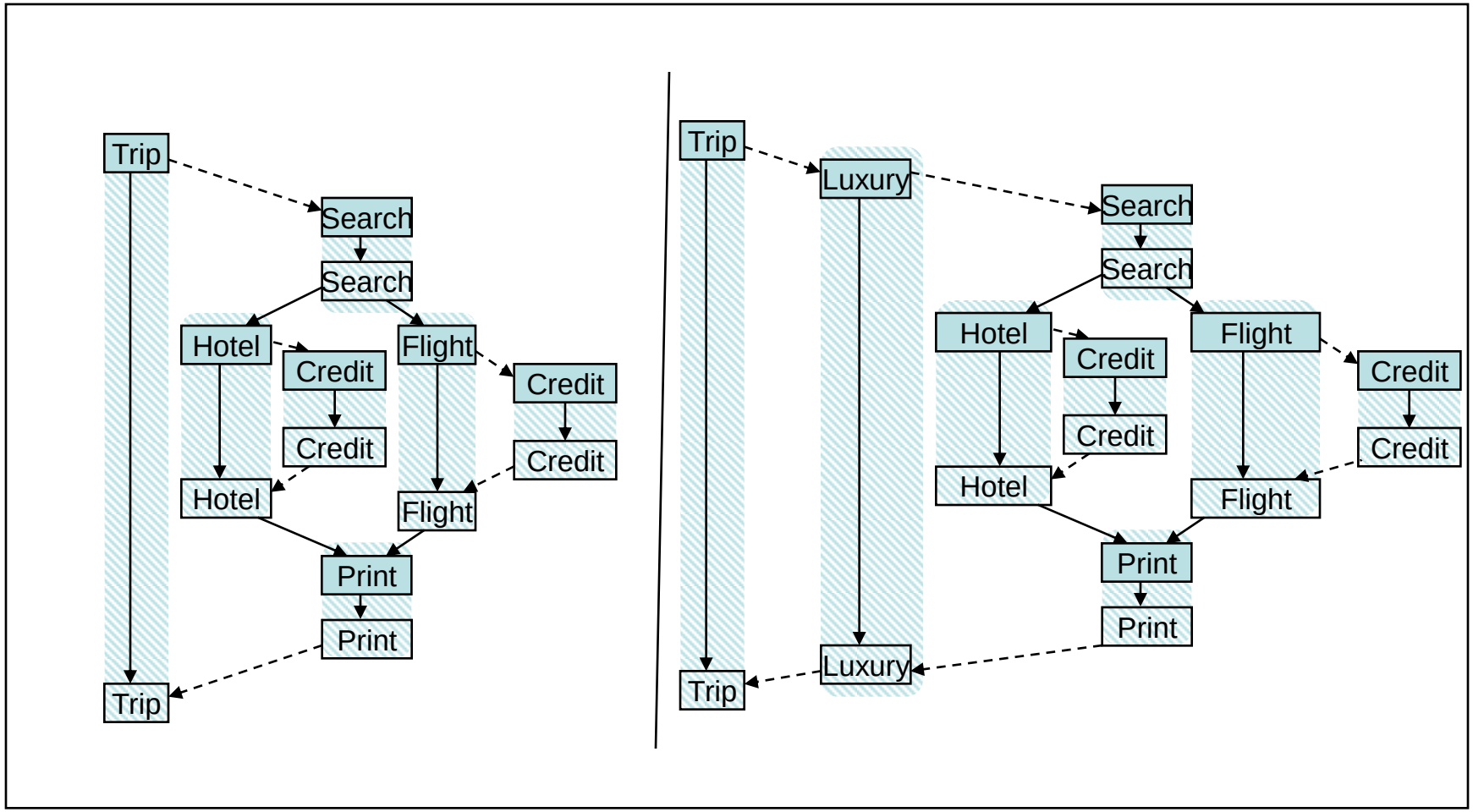
Example BP


Trip
Trip
Luxury
Luxury
Search
Search
Hotel
Hotel
Credit1
Credit1
Flight
Flight
Credit1
Credit1
Print
Print
Search
Search
LuxHotel
LuxHotel
Credit2
Credit2
LuxFlight
LuxFlight
Credit2
Credit2
Print
Print

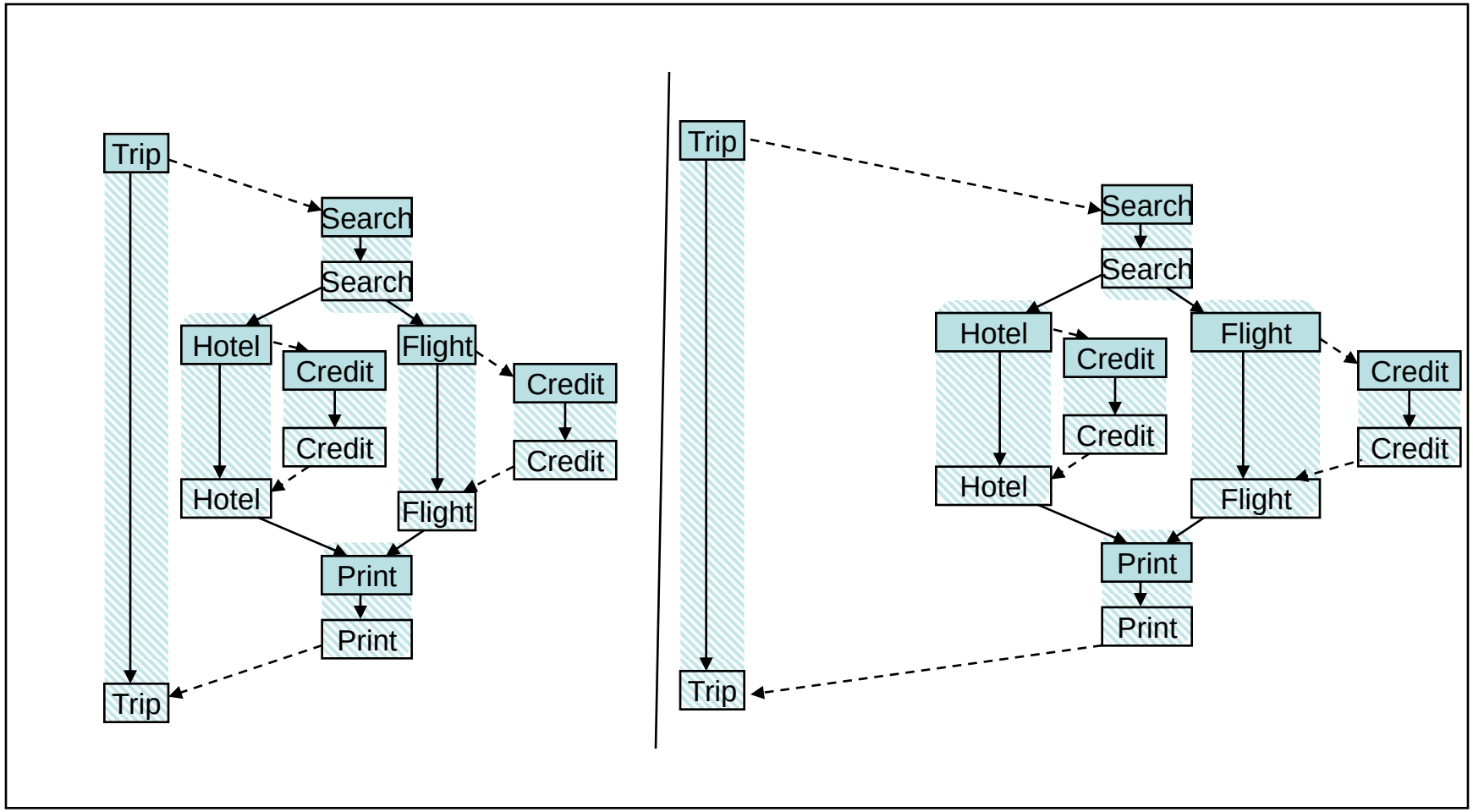
Naïve Traces



Semi-Naïve Traces



Selective Traces



Lets talk about (top-k) queries

- Given a **partial trace**, what is its most likely origin?
- Or, more generally,
  given a **pattern (query)** of partial traces, what are the most likely origins of partial traces of this pattern ?
- Good news:
  Many of the query evaluation algorithms extend to this context (even without knowing the tracing system…)

Outline

- Introduction & Motivation
- Querying Future
- Querying Past
- Related and Future work

(Small subset of) Related work

- (Probabilistic) Recursive State Machines with temporal logic as query language
- Probabilistic Relational DBs
- Probabilistic XML
- Graph grammars with MSO (or FO) as query language
- BP and Web applications mining

Future work

- Practical applications:
  - Web-sites design
  - On-line advertisements
  - Improved business logic
- Enriched Query Language
  - Joins
  - Data values
- Optimization
- Inference of specs/probability distributions

(= Thanks !)