The Raindrop Engine: Continuous Query Processing at WPI

Elke A. Rundensteiner
Database Systems Research Lab, WPI
2003

This talk is based on joint work with students at DSRG lab.

Invited Talk at Database Seminar Series @ Univ. of Waterloo, Canada.

Monitoring Applications

- Monitor troop movements during combat and warn when soldiers veer off course
- Send alert when patient's vital signs begin to deteriorate
- Monitor incoming news feeds to see stories on "Iraq"
- Scour network traffic logs looking for intruders (this is our focus in Raindrop)

Properties of Monitoring Applications

- Queries and monitors run continuously, possibly unending
- Applications have varying service preferences:
  - Patient monitoring only wants freshest data
  - Remote sensors have limited memory
  - News service wishes maximal throughput
  - Taking 60 seconds to process vital signs and sound an alert may be too long

Properties of Streaming Data

- Possibly never ending stream of data
- Unpredictable arrival patterns:
  - Network congestion
  - Weather (for external sensors)
  - Sensor moves out of range

DBMS Approach to Continuous Queries

- Insert each new tuple into the database, encode queries as triggers [MWA03]
- Problems:
  - High overhead with inserts [CCC02]
  - Triggers do not scale well [CCC02]
  - Uses static optimization and execution strategies that cannot adapt to unpredictable streams
  - System is less utilized if data streams arrive slowly
  - No means to input application service requirements

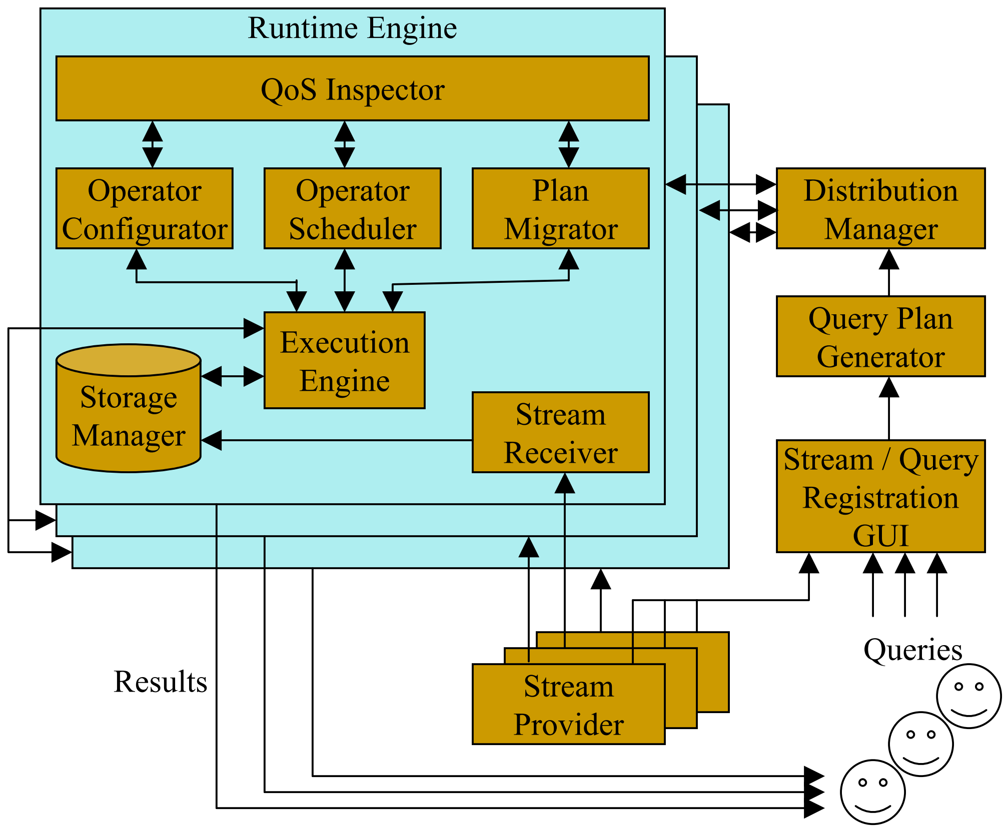
Hence, New Class of Query Systems

- CQ Systems emerged recently (Aurora, Stream, NiagaraCQ, Telegraph, et al.)
- Generally work as follows:
  - System subscribes to some streams
  - End users issued continuous queries against streams
  - System processes queries continuously
  - System returns the results to the user as a stream
- All CQ systems use adaptive techniques to cope with unpredictability of streams and environment

Overview of Adaptive Techniques in CQ Systems

Research Work	**Technique(s)**	**Goal**
Aurora [CCC02]	Load shedding, batch tuple processing to reduce context switching	Maintain high quality of service
STREAM [MWA03]	Adaptive scheduling algorithm (Chain) [BBM03], Load shedding	Minimize memory requirements during periods of bursty arrival
NiagaraCQ [CDT00]	Generate near-optimal query plans for multiple queries	Efficiently share computation between multiple queries, highly scalable system, maximize output rate
Eddies [AH00] (Telegraph)	Dynamically route tuples among Joins	Keep system constantly busy, improve throughput
XJoin [UF00]	Break Join into 3 stages and make use of memory and disk storage	Keep Join and system running at full capacity at all times
[UF01]	Schedule streams with the highest rate	Maximize throughput to clients
Tukwila [IFF99] [UF98]	Reorganize query plans on the fly by using using synchronization packets to tell operators to finish up their current work.	Improve ill-performing query plans

The WPI Stream Project: Raindrop



Raindrop Project Goal

- Our goal is to bring adaptability into all components of the system;
- Without reniging on the requested accuracy of answer;
- While optimizing for a given set of quality of service requirements.

- Quality of service requirements are multi-faceted, such as memory usage, timeliness, output rate, etc., and can even be modified at run-time.
- Plus, we support self-inspection of components tp observe the effect of their actions on environment, and respond appropriately.

Adaption in All Components

- Scalable Query Operators (Punctuations)
- Cooperative Plan Optimization
- Adaptive Operator Scheduling
- On-line Query Plan Migration
- Distributed Plan Execution

Adaption in All Components

- Scalable Query Operators (Punctuations)
  - Adapt and select among tasks such as memory purging, stream reading, memory-to-disk shuffling, punctuation propagation, etc.
- Cooperative Plan Optimization
  - Operators request and selectively provide punctuations (meta-knowledge) to improve their performance
- Adaptive Operator Scheduling
  - Selector scores alternate scheduling algorithm based on their effect on QoS requirements, and selects candidate.
- On-line Query Plan Migration
  - On-line plan restructuring and then online migration to the new plan even for stateful operators.
- Distributed Plan Execution
  - Adaptively distribute computations across multiple machines to optimize QoS requirements without information loss

Topics Studied in Raindrop Project

- PLUS, Bring XML into Stream Engine (!?)
- Scalable Query Operators (Punctuations)
- Cooperative Plan Optimization
- Adaptive Operator Scheduling
- On-line Query Plan Migration
- Distributed Plan Execution

Raindrop

- Let’s now focus on putting XML into Raindrop, as each of the other topics is a separate story to be told at another time.

PART II: XQueries on XML Streams (Automaton Meets Algebra)

Based on CIKM'03 and ER'03

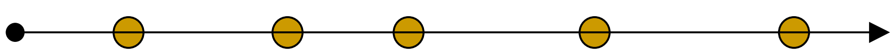
Joint work with Hong Su and Jinhui Jian

What's Special for XML Stream Processing?

```
<Biditems>
    <book year="2001">
        <title>Dream Catcher</title>
        <author><last>King</last><first>S.</first></author>
        <publisher>Bt Bound </publisher>
        <price> 30 </initial>
    </book>
…
```

Token-by-Token access manner

<biditems> <book> <title> Dream Catcher </title> …



timeline

Token: not a direct counterpart of a tuple

year	title	last	first	publisher	price
2001	Dream	King	S.	Bt Bound	30

Pattern retrieval + *Filtering* + *Restructuring*

```
FOR $b in stream(biditems.xml) //book
LET $p := $b/price
    $t := $b/title
WHERE $p < 20
Return <Inexpensive> $t </Inexpensive>
```

Pattern Retrieval on Token Streams

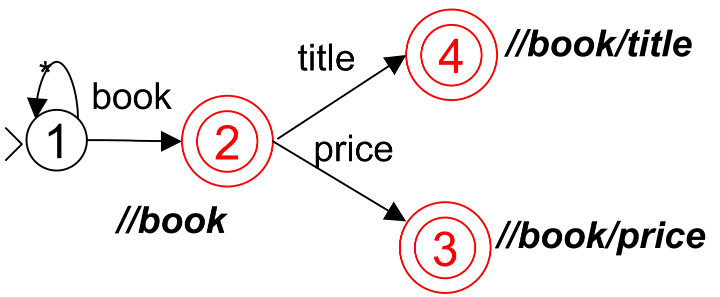
Two Computation Paradigms

- Automata-based [yfilter02, xscan01, xsm02, xsq03, xpush03…]
- Algebraic [niagara00, …]

This Raindrop framework intends to integrate both paradigms into one

Automata-Based Paradigm

```
FOR $b in stream(biditems.xml) //book
LET $p := $b/price
      $t := $b/title
WHERE $p < 20
Return <Inexpensive> $t </Inexpensive>
```



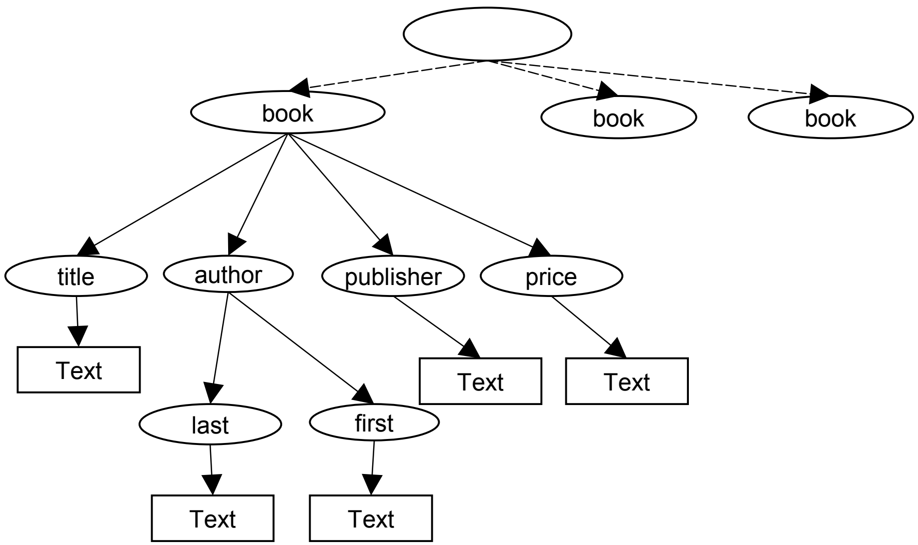
Auxiliary structures for:

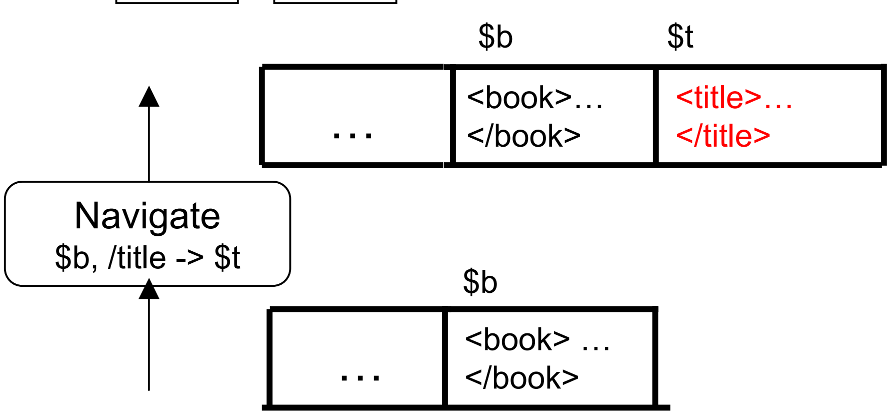
1. Buffering data
2. Filtering
3. Restructuring

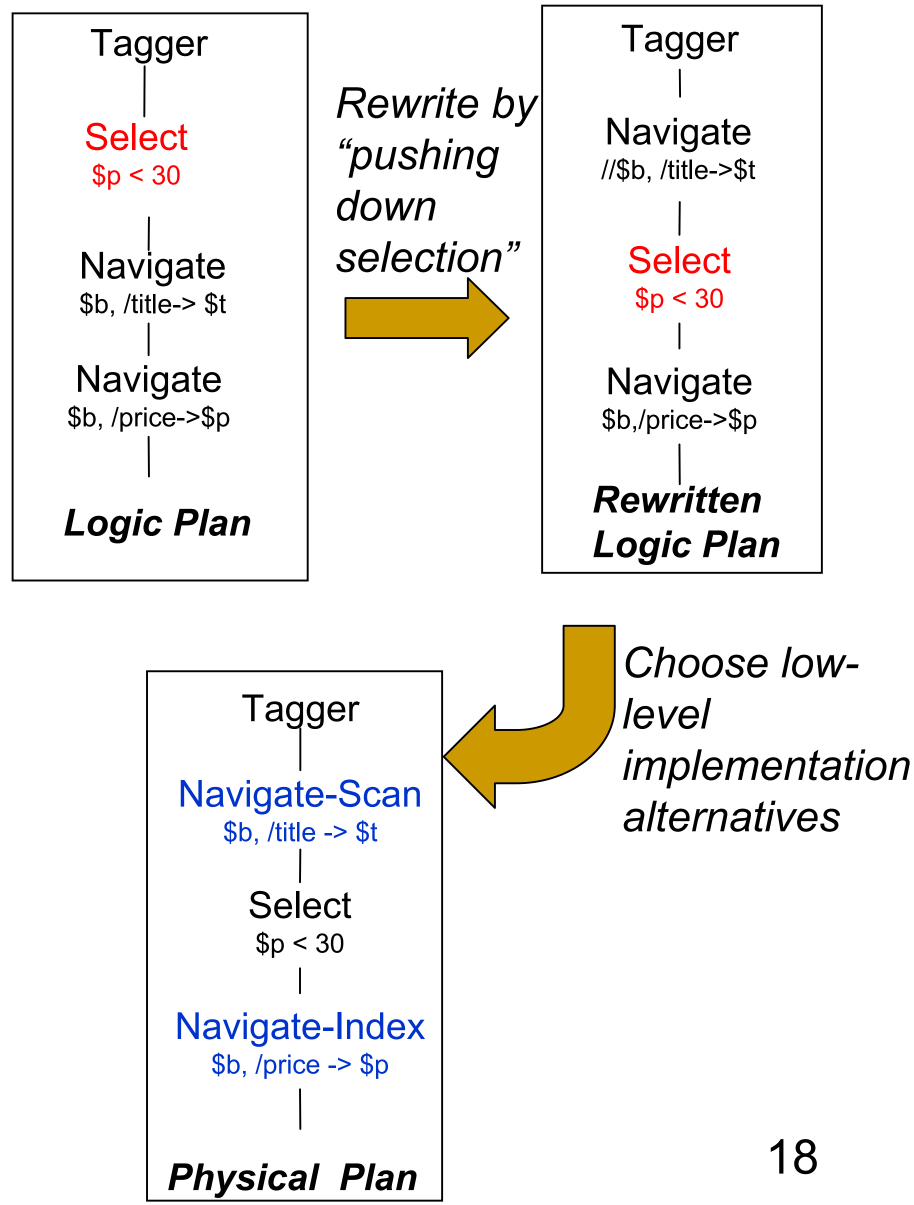
…

Algebraic Computation

FOR $b in stream(biditems.xml) //book
LET $p := $b/price
$t := $b/title
WHERE $p < 20
Return <Inexpensive> $t </Inexpensive>







Observations

Automata Paradigm	*Algebra Paradigm*
Good for pattern retrieval on tokens	Does not support token inputs
Need patches for filtering and restructuring	Good for filtering and restructuring
Present all details on same low level	Support multiple descriptive levels (declarative->procedural)
Little studied as query processing paradigm	Well studied as query process paradigm

Either paradigm has deficiencies

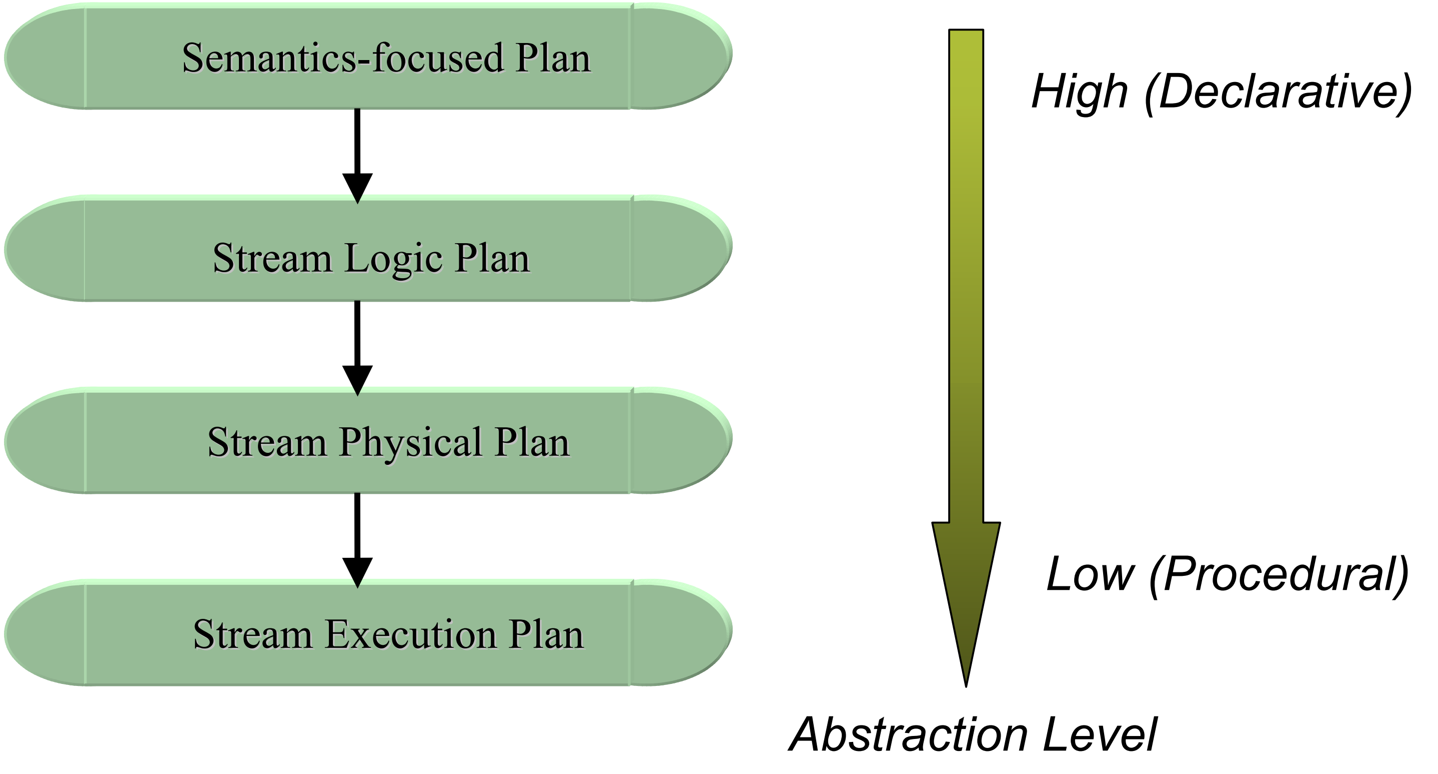
Both paradigms complement each other

How to Integrate Two Paradigms

How to Integrate Two Models?

- Design choices
  - Extend algebraic paradigm to support automata?
  - Extend automata paradigm to support algebra?
  - Come up with completely new paradigm?
- Extend algebraic paradigm to support automata
  - Practical
    - Reuse & extend existing algebraic query processing engines
  - Natural
    - Present details of automata computation at low level
    - Present semantics of automata computation (target patterns) at high level

Raindrop: Four-Level Framework



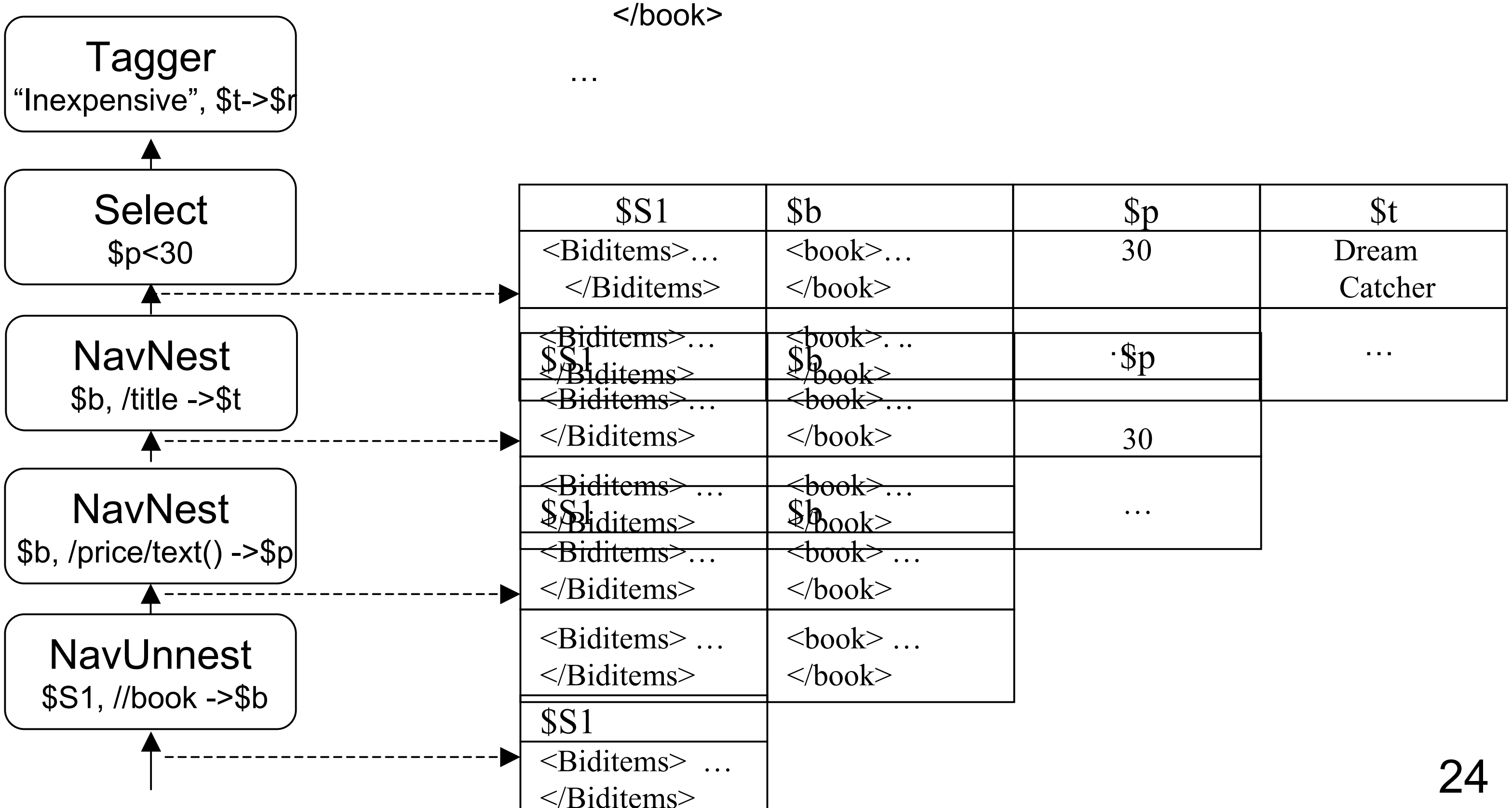
Level I: Semantics-focused Plan [Rainbow-ZPR02]

- Express query semantics regardless of stored or stream input sources
- Reuse existing techniques for stored XML processing
  - Query parser
  - Initial plan constructor
  - Rewriting optimization
    - Decorrelation
    - Selection push down
    - ...

Example Semantics-focused Plan

```
FOR $b in stream(biditems.xml) //book
LET $p := $b/price
    $t := $b/title
WHERE $p < 20
Return <Inexpensive> $t </Inexpensive>
```

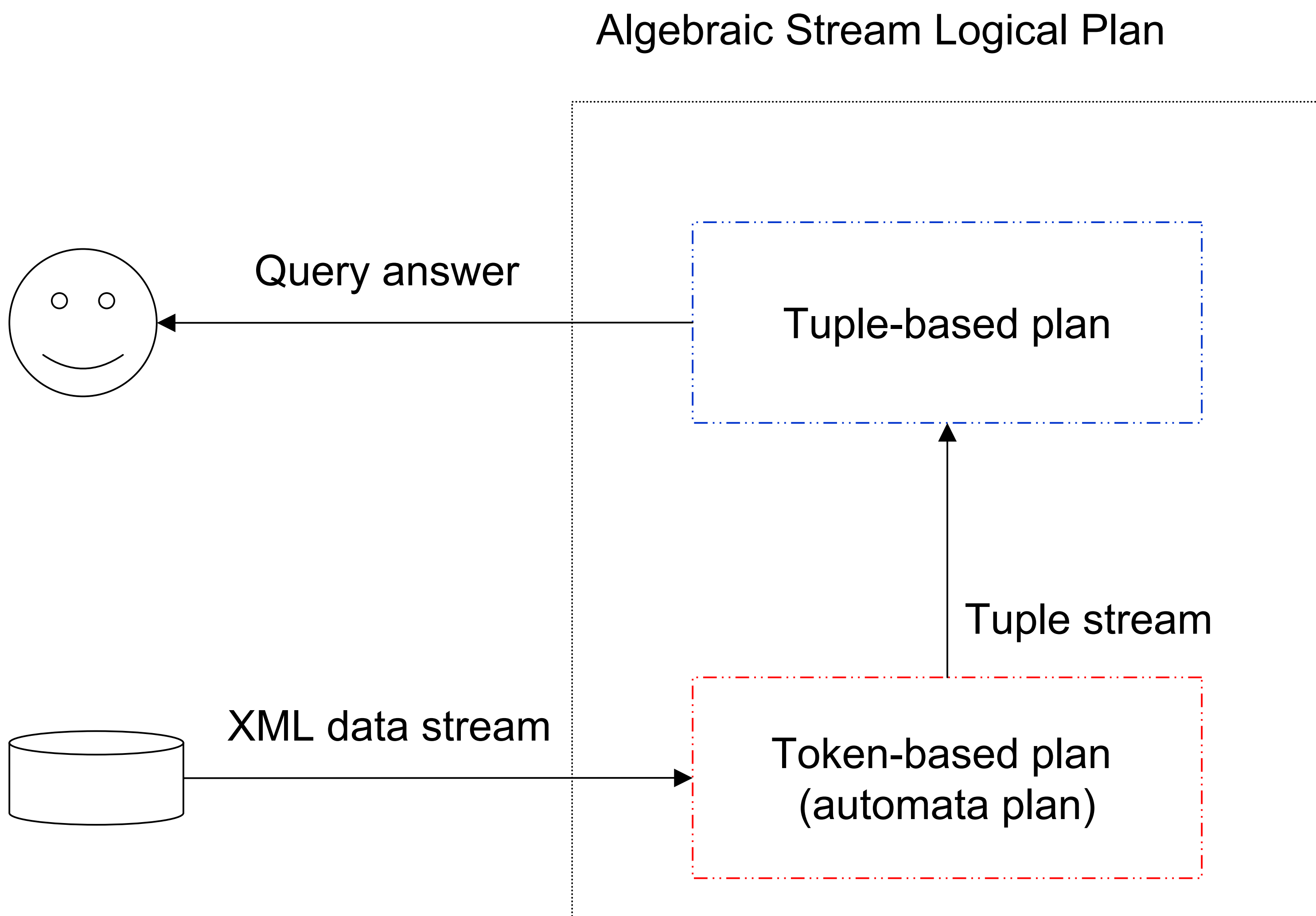
```
<Biditems>
    <book year="2001">
      <title>Dream Catcher</title>
      <author><last>King</last><first>S.</first></author>
      <publisher>Bt Bound </publisher>
      <price> 30 </initial>
    </book>
  …
```



Level II: Stream Logical Plan

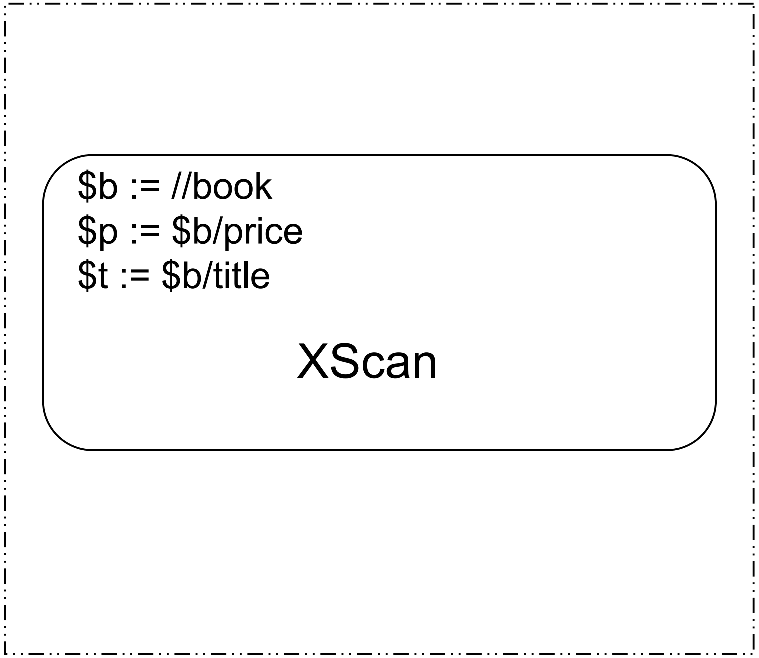
- Extend semantics-focused plan to accommodate tokenized stream inputs
  - New input data format:
    - *contextualized tokens*
  - New operators:
    - *StreamSource, Nav, ExtractUnnest, ExtractNest, StructuralJoin*
  - New rewrite rules:
    - *Push-into-Automata*

One Uniform Algebraic View

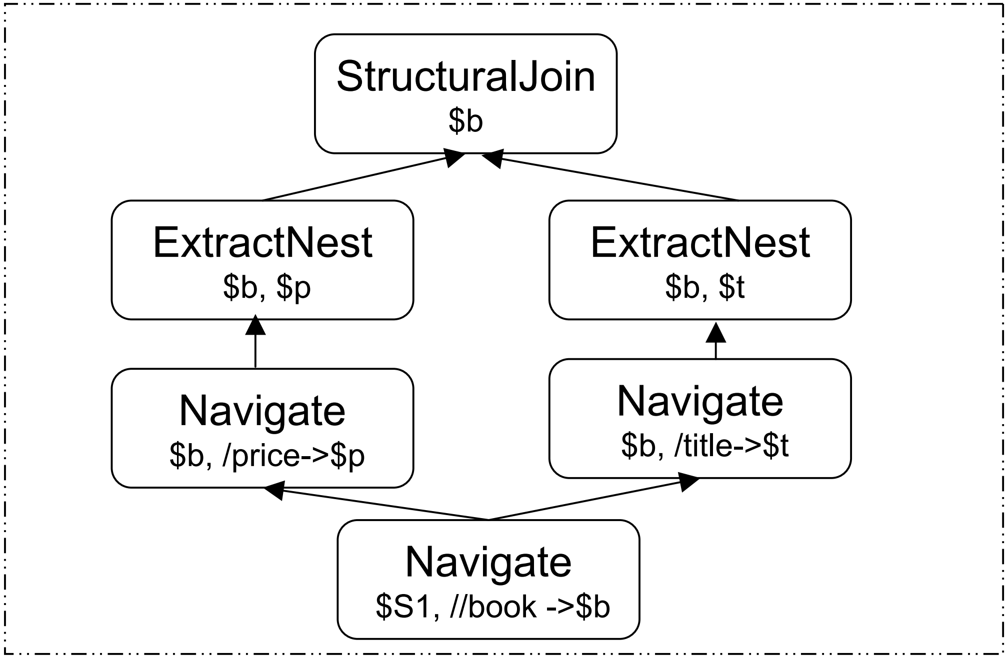


Modeling the Automata in Algebraic Plan: Black Box[XScan01] vs. White Box

```
FOR $b in stream(biditems.xml) //book
LET $p := $b/price
    $t := $b/title
WHERE $p < 20
Return <Inexpensive> $t </Inexpensive>
```



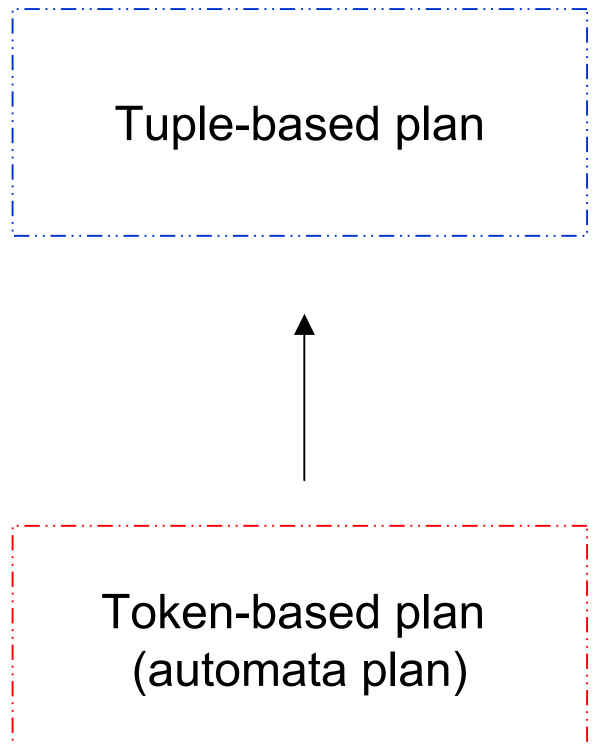
Black Box



White Box

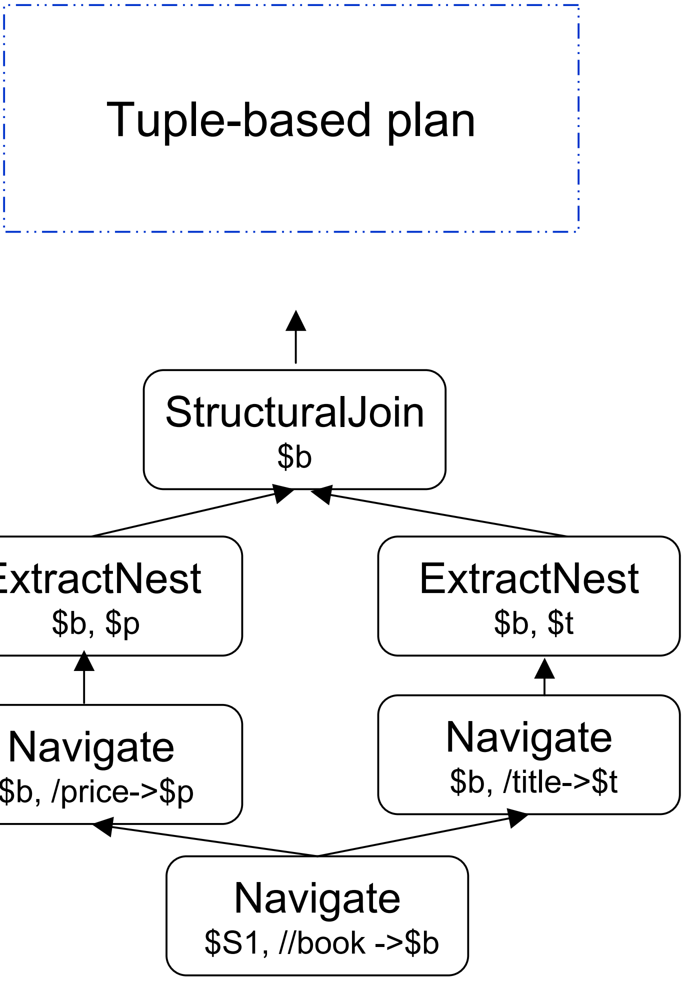
Example Uniform Algebraic Plan

```
FOR $b in stream(biditems.xml) //book
LET $p := $b/price
        $t := $b/title
WHERE $p < 30
Return <Inexpensive> $t </Inexpensive>
```



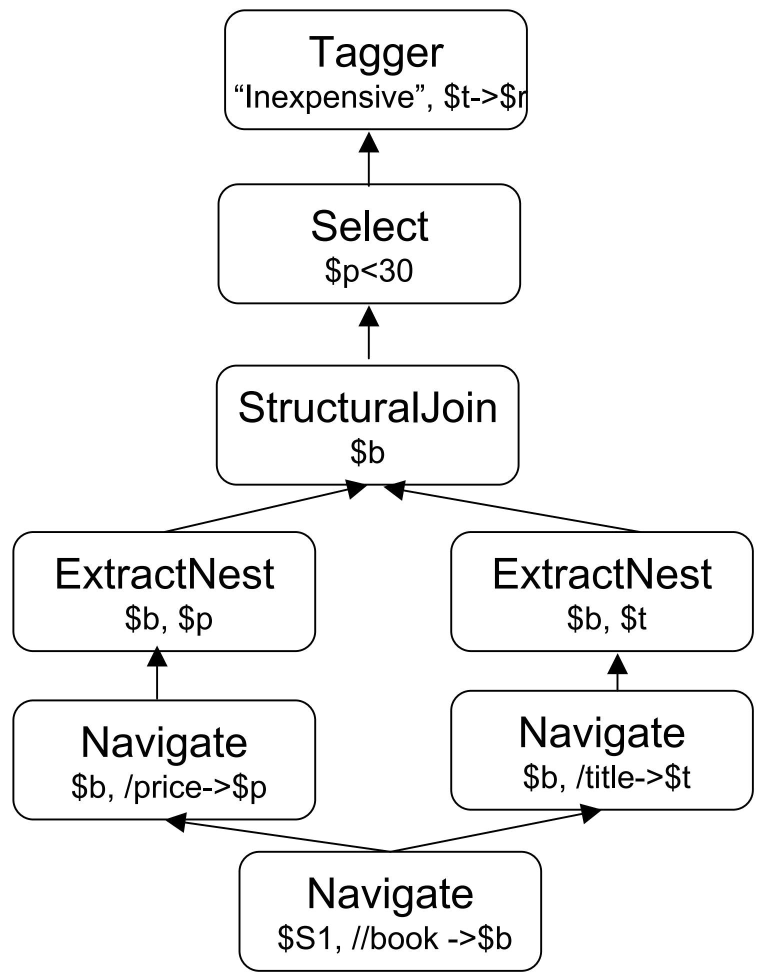
Example Uniform Algebraic Plan

FOR $b in stream(biditems.xml) //book
LET $p := $b/price
    $t := $b/title
WHERE $p < 30
Return <Inexpensive> $t </Inexpensive>

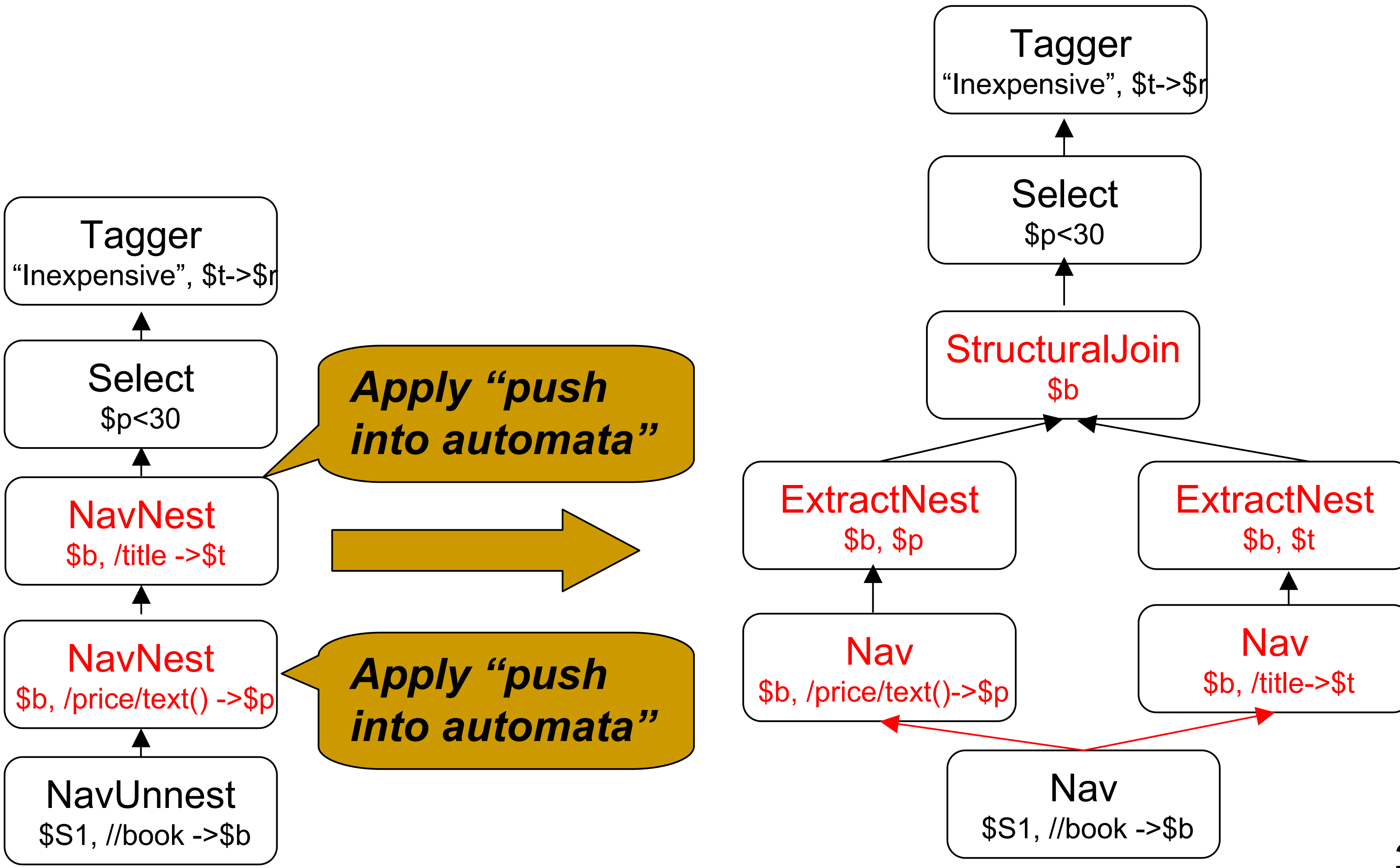


Example Uniform Algebraic Plan

FOR $b in stream(biditems.xml) //book
LET $p := $b/price
$t := $b/title
WHERE $p < 30
Return <Inexpensive> $t </Inexpensive>



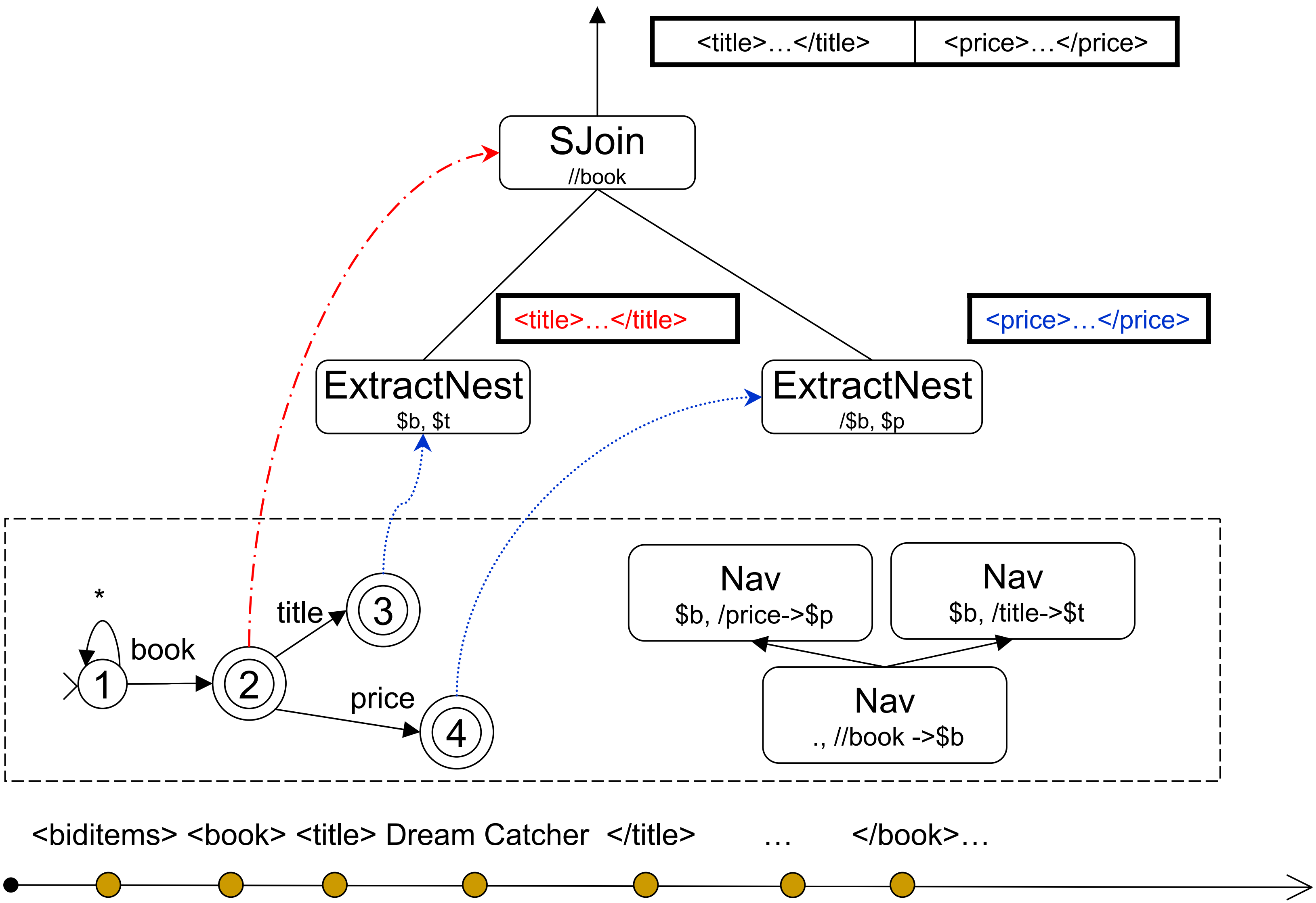
From Semantics-focused Plan to Stream Logical Plan



Level III: Stream Physical Plan

- For each stream logical operator, define how to generate outputs when given some inputs
  - Multiple physical implementations may be provided for a single logical operator
  - Automata details of some physical implementation are exposed at this level
    - *Nav, ExtractNest, ExtractUnnest, Structural Join*

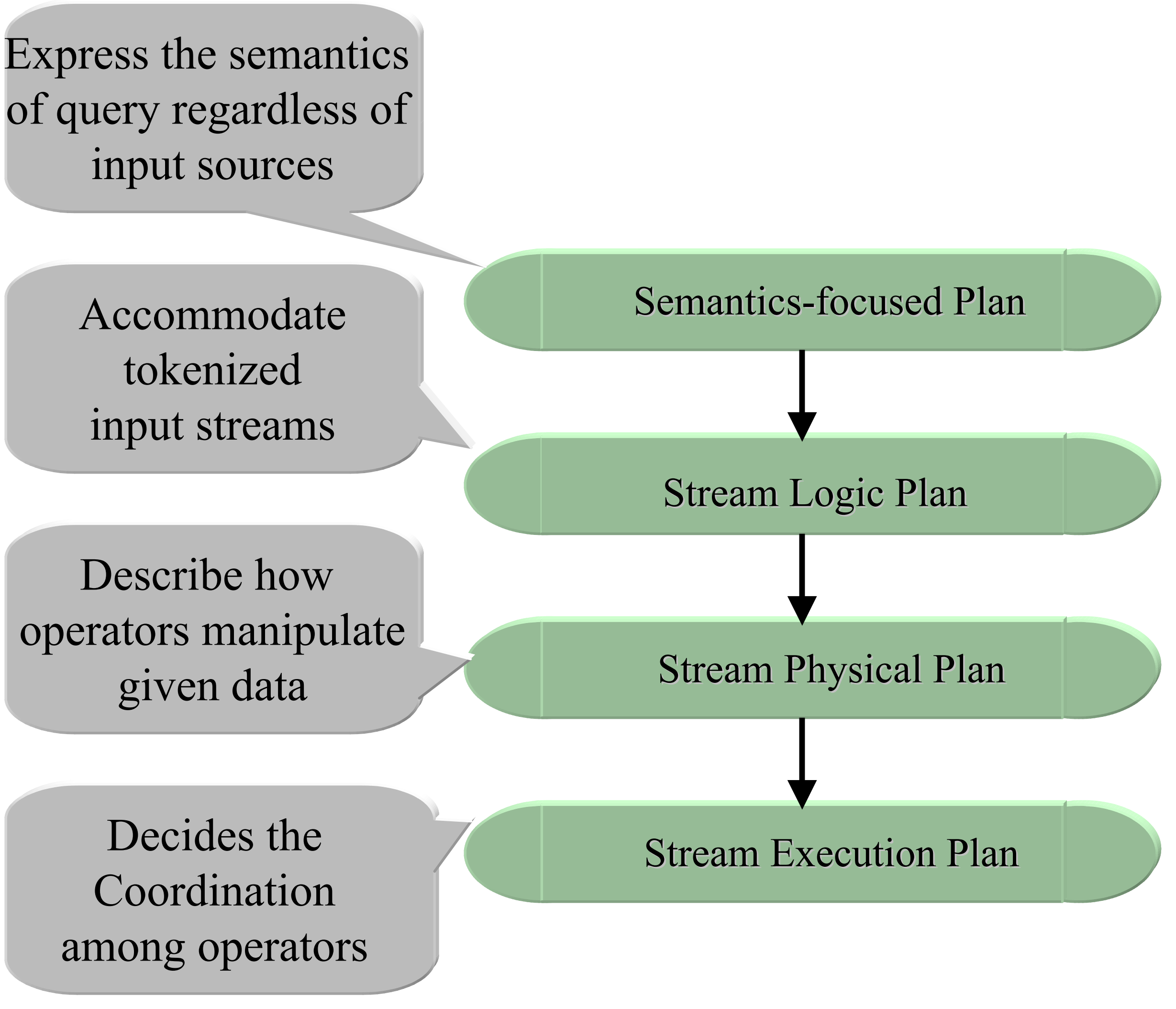
One Implementation of Extract/Structural Join



Level IV: Stream Execution Plan

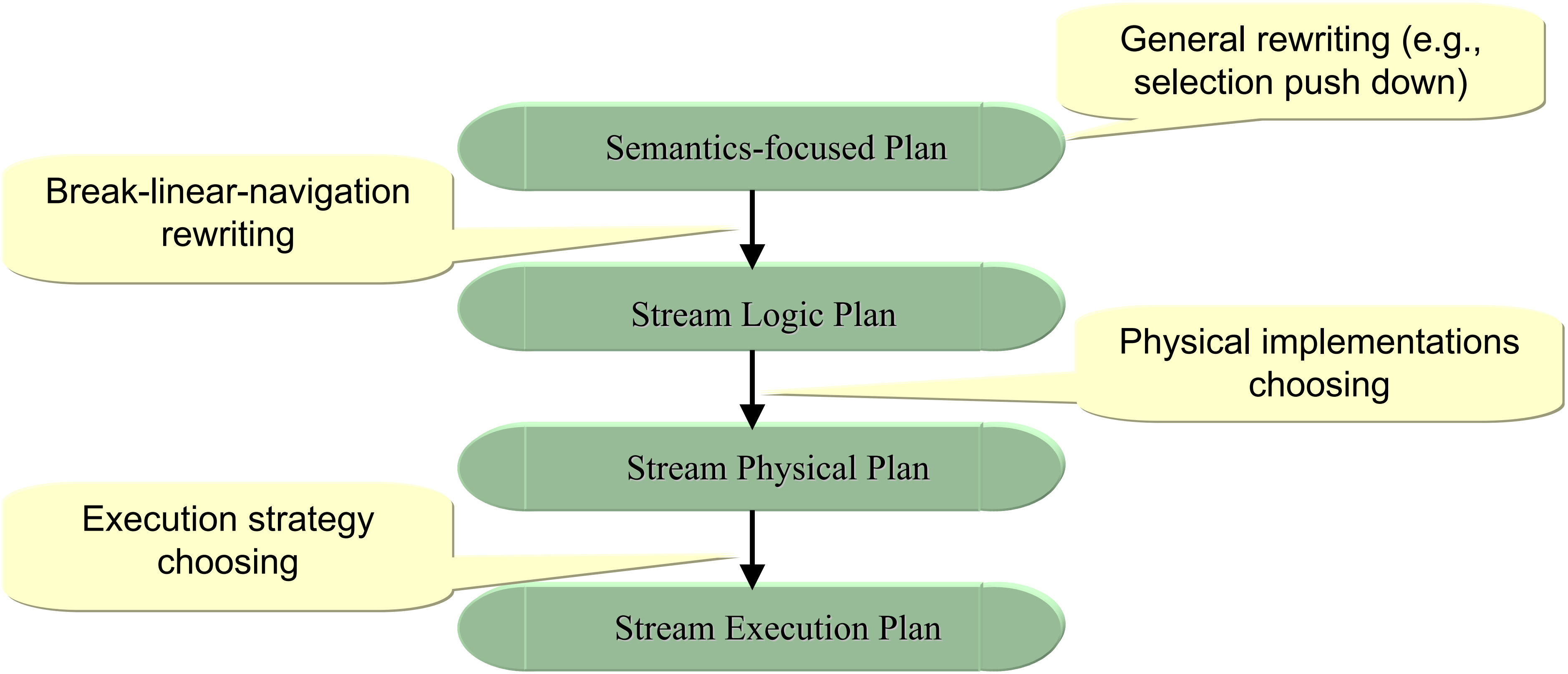
- Describe coordination between operators regarding when to fetch the inputs
  - When input operator generates one output tuple
  - When input operator generates a batch
  - When a time period has elapsed
  - ...
- Potentially unstable data arrival rate in stream makes fixed scheduling strategy unsuitable
  - Delayed data under scheduling may stall engine
  - Bursty data not under scheduling may cause overflow

Raindrop: Four-Level Framework (Recap)

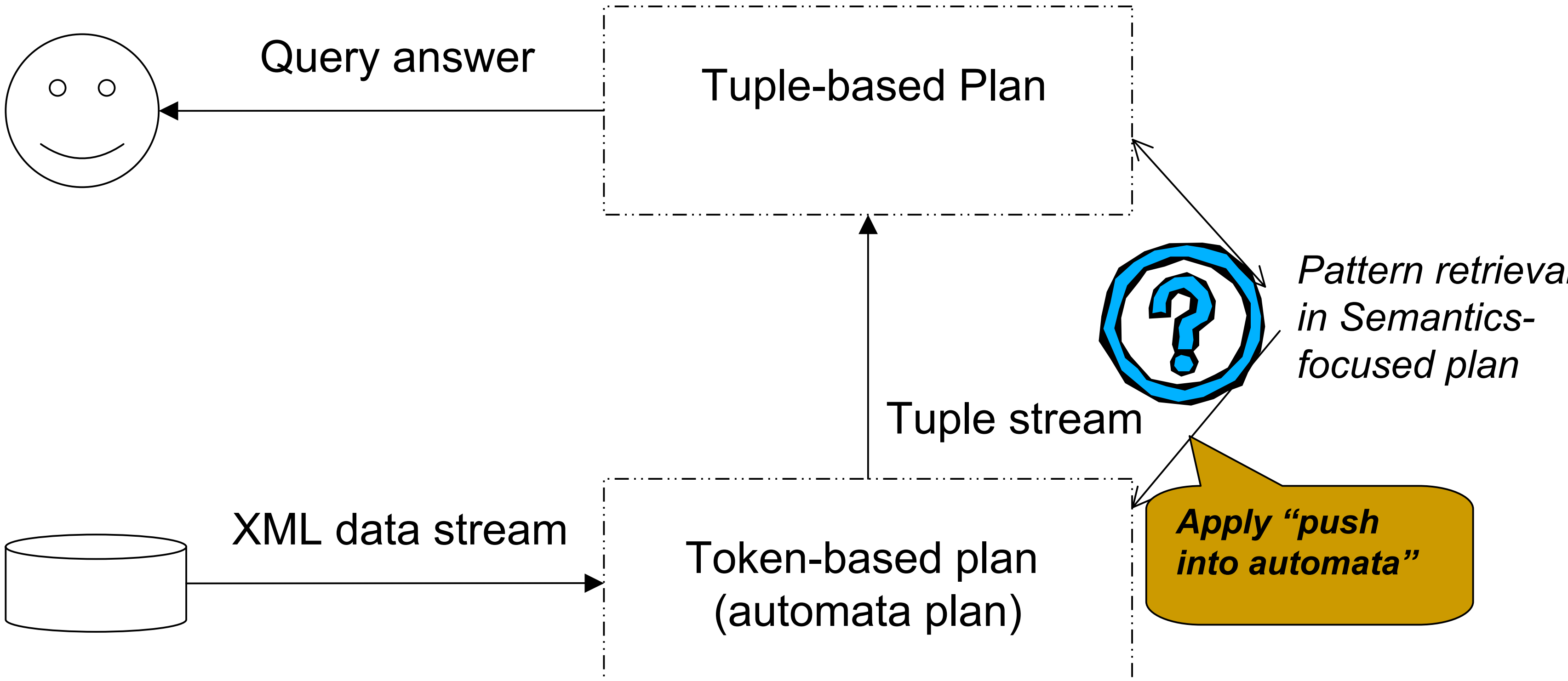


Optimization Opportunities

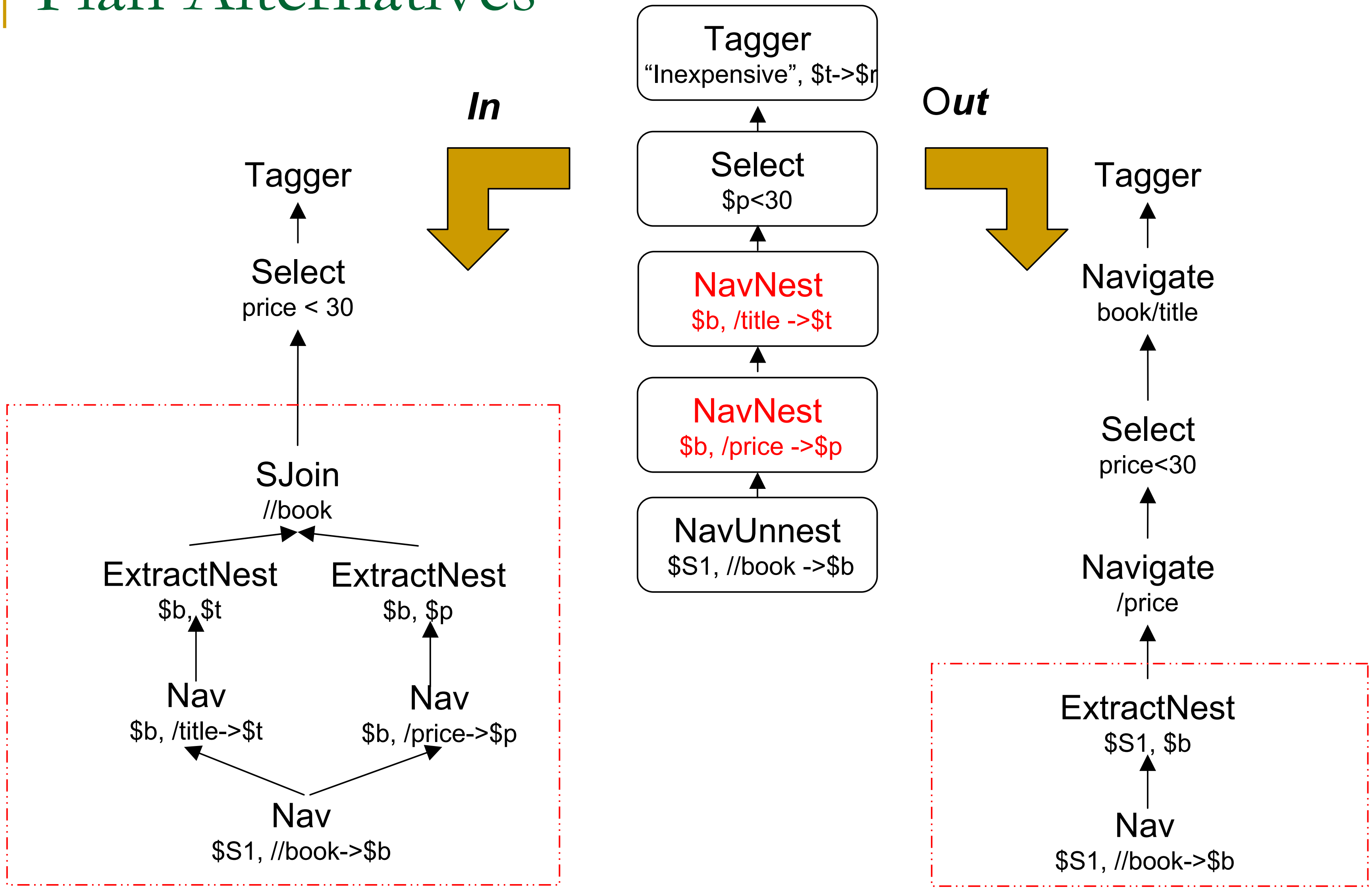
Optimization Opportunities



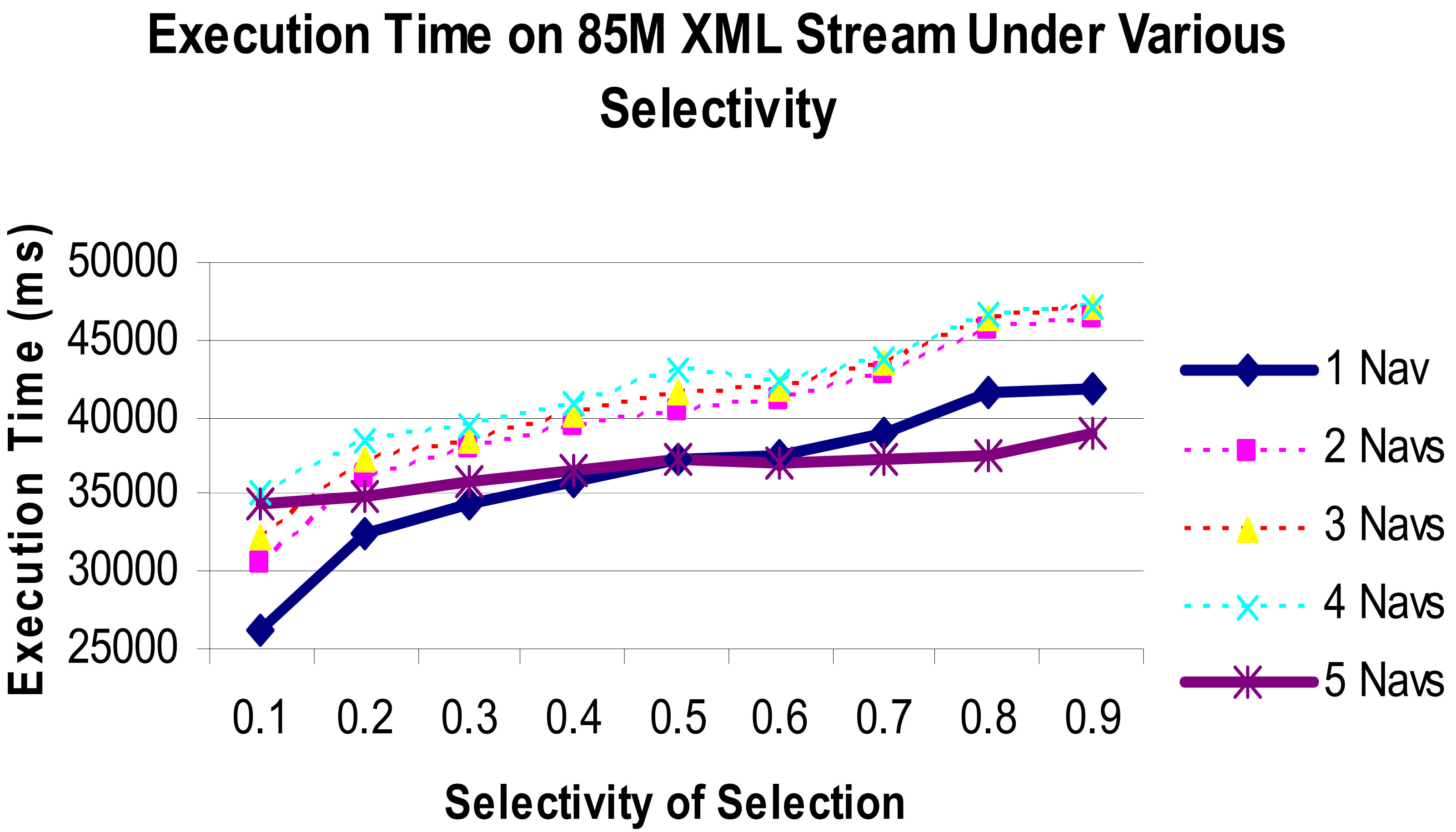
From Semantics-focused to Stream Logical Plan: In or Out?



Plan Alternatives



Experimentation Results



Contributions Thus Far

- Combined automata and algebra based paradigms into one uniform algebraic paradigm
- Provided four layers in algebraic paradigm
  - Query semantics expressed at high layer
  - Automata computation on streams hidden at low layer
- Supported optimization at an iterative manner (from high abstraction level to low abstraction level)
- Illustrated enriched optimization opportunities by experiments

Automaton Meets Algebra: On-Going Issues To be Tackled

- Exploit XML schema constraints for query optimization
- Costing/query optimization of plans
- On-the-fly migration into/out of automaton
- Physical implementation strategies of operators
- Load-shedding from an automaton

Plus, Other Topics in Raindrop

- Bring XML into Stream Engine
- Scalable Query Operators (Punctuations)
- Cooperative Plan Optimization
- Adaptive Operator Scheduling
- On-line Query Plan Migration
- Distributed Plan Execution

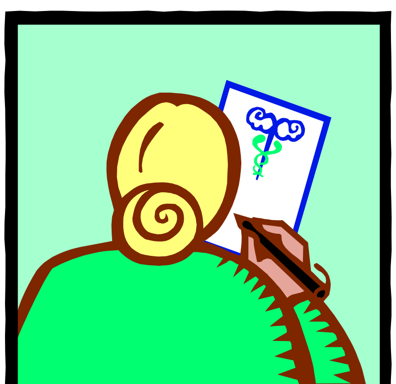
To be told on another day …

Overall Blizz . . .

- Many interesting problems arise in this new stream context
- There is room for lot's of fun research

http://davis.wpi.edu/dsrg/raindrop/

Here you can find Project Overview, Publications, and Talks



Email: raindrop@cs.wpi.edu