Learning to Optimize Plan Execution in Information Agents

Craig A. Knoblock

University of Southern California

Acknowledgements

- Electric Elves
  - Jose Luis Ambite
  - Maria Muslea
  - Hans Chalupsky
  - Yolanda Gil
  - Jean Oh
  - David V. Pynadath
  - Thomas A. Russ
  - Milind Tambe
- Theseus Agent Execution
  - Greg Barish
  - Steve Minton
  - Maria Muslea
- Speculative Execution
  - Greg Barish
- Funding
  - DARPA
  - AFOSR
  - NSF

Introduction

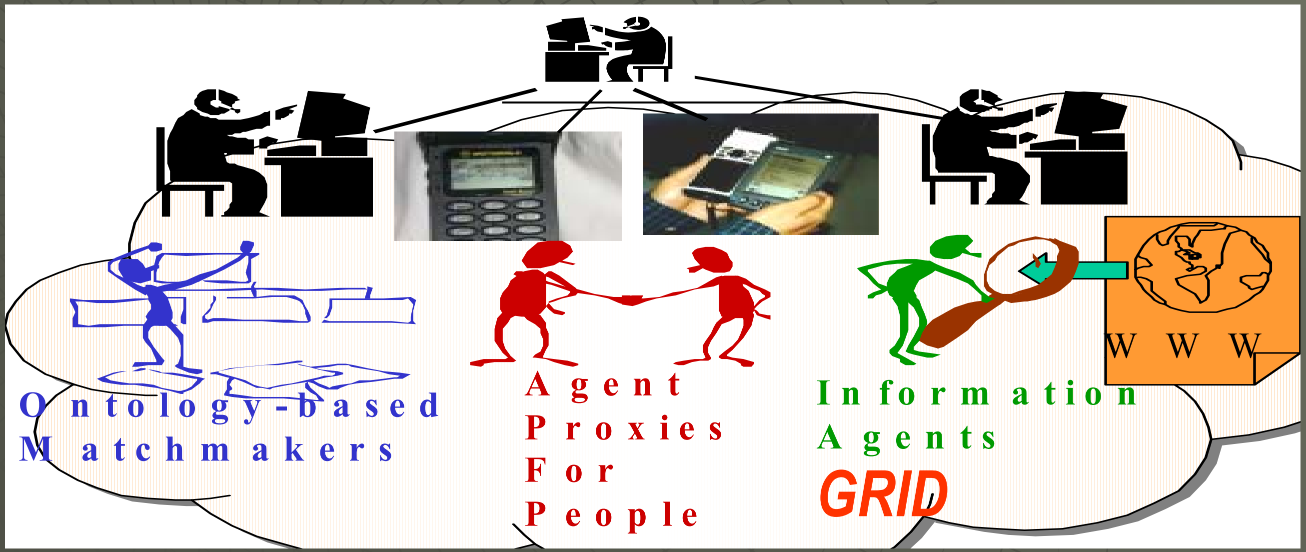
- The Web is a tremendous resource, but designed for browsing
  - Sites provide limited capabilities for personalization
  - Few sites are designed to be integrated with others
- Goal: Develop technology to rapidly construct personal software agents
  - Build agents that can perform retrieval, integration, and monitoring tasks on any online source

Outline

1. Motivating Application: The Electric Elves
2. Efficiently executing agent plans
3. Speculative plan execution
4. Value prediction for speculative execution
5. Related Work
6. Conclusions

Electric Elves Project
[Chalupsky et al, 2001]

- Elves project goal: Apply agent technology to support human organizations
  - Develop software agents that automate routine tasks
  - Enable software agents and humans to work together
  - Support coordination of tasks
- Applications: Office Elves and Travel Elves



Agents for Monitoring Travel
[Ambite et al, 2002]

- Travel Elves created as an application of the Electric Elves
- Given travel itinerary, generates set of agents for anticipating travel-related failures and opportunities:
  - Price changes
  - Schedule changes
  - Flight delays & cancellations
  - Earlier and close connections
  - Finding the closest restaurant given GPS coordinates

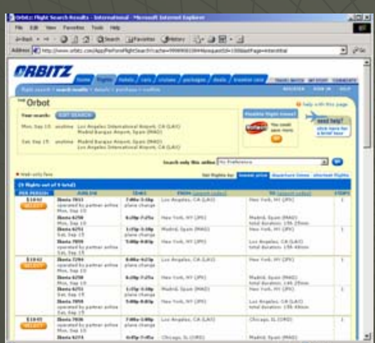
Monitoring Travel Plans

Monitoring Tasks

Monitor Flight Status	(•) Monitor Flights / ( ) Stop Monitoring	7038128516 — Notify Hotel (Fax)	7034948462 — Notify Car Rental Counter (Fax)		
Status	Active — Outbound flight 1	Active — Outbound flight 2	Active — Inbound flight 1	Active — Inbound flight 2	
Monitor Flight Schedule	(•) Monitor Schedule / ( ) Stop Monitoring	Active — Status			
Monitor Earlier Flights	(•) Monitor Earlier Flights / ( ) Stop Monitoring	Active — Status			
Monitor Connecting Flights	(•) Monitor Connecting Flights / ( ) Stop Monitoring	Active — Status (Outbound)	Active — Status (Inbound)		
Monitor Airfare	Decrease only — Mode	(•) Monitor Airfare / ( ) Stop Monitoring — Airfare	Active — Status		

Agents Deployed to Monitor Travel Itinerary

Travel Itinerary



Flight Prices & Schedules

Flight Status

Weather

Restaurants

Monitoring Agents

- Flight-Status Agent:
  - Flight delayed message:

    Your United Airlines flight 190 has been delayed.

    It was originally scheduled to depart at 11:45 AM and is now scheduled to depart at 12:30 PM.

    The new arrival time is 7:59 PM.
  - Flight cancelled message:

    Your Delta Air Lines flight 200 has been cancelled.
  - Fax to hotel message:

    Attention: Registration Desk

    I am sending this message on behalf of David Pynadath, who has a reservation at your hotel. David Pynadath is on United Airlines 190, which is now scheduled to arrive at IAD at 7:59 PM. Since the flight will be arriving late, I would like to request that you indicate this in the reservation so that the room is not given away.

Monitoring Agents

- Airfare Agent: Airfare dropped message

```
The airfare for your American Airlines itinerary
(IAD - LAX) dropped to $281.
```

- Earlier-Flight Agent: Earlier flights message

```
The status of your currently scheduled flight is:
# 190 LAX (11:45 AM) - IAD (7:29 PM) 45 minutes Late
If you would like to return earlier, the following
United Airlines flights will arrive earlier than your
scheduled flights:
# 946 LAX (8:31 AM) - IAD (3:35 PM) 11 minutes Late
--------
# 388 LAX (9:25 AM) - DEN (12:25 PM) 10 minutes Late
# 1534 DEN (1:20 PM) - IAD (6:06 PM) On Time
```

Outline

1. Motivating Application: The Electric Elves
2. Efficiently executing agent plans
3. Speculative plan execution
4. Value prediction for speculative execution
5. Related Work
6. Conclusions

Efficiently Executing Agent Plans

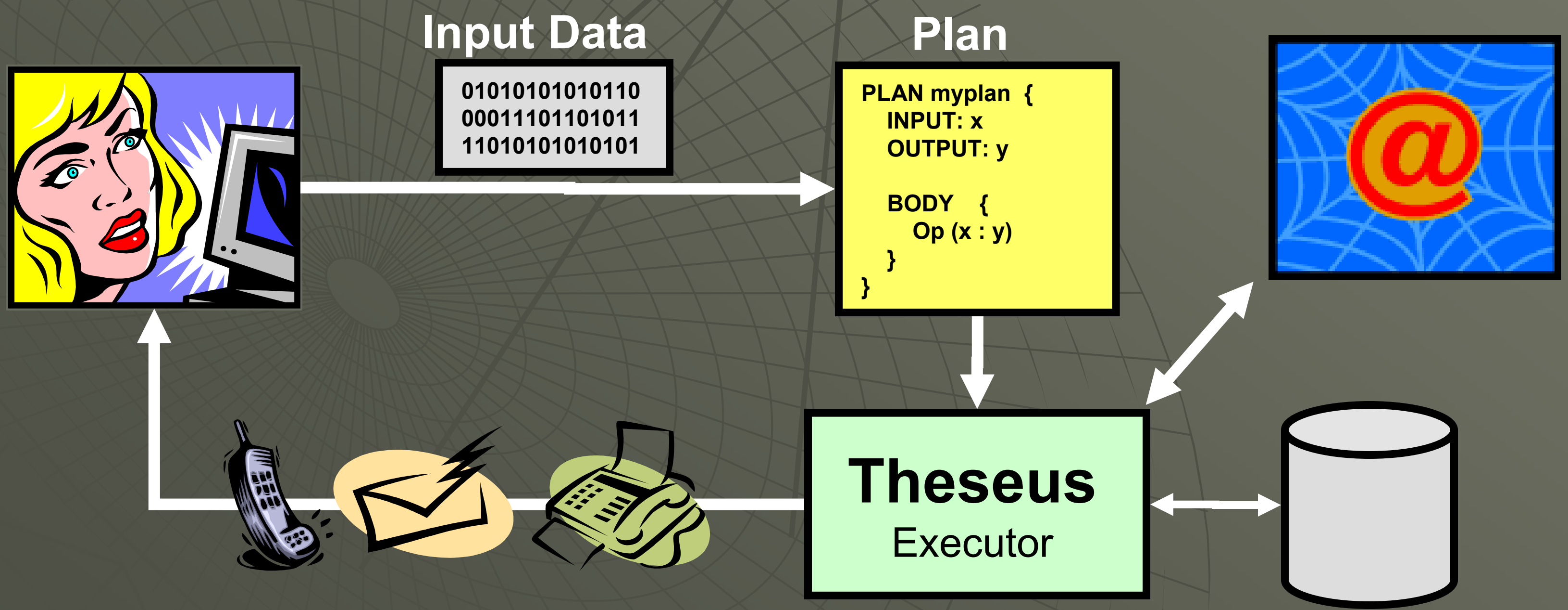
- **Problem**
  - Information gathering may involve accessing and integrating data from many sources
  - Total time to execute these plans may be large

- **Why?**
  - Slow remote sources
  - Unpredictable network latencies
  - Binding patterns
    - Source cannot be queried until a previous query has been answered
  - Result: execution is often I/O-bound

Theseus Agent Execution System

[Barish & Knoblock, JAIR'05]

- **Plan language** and **execution system** for Web-based information integration
  - Expressive enough for monitoring a variety of sources
  - Efficient enough for real-time monitoring



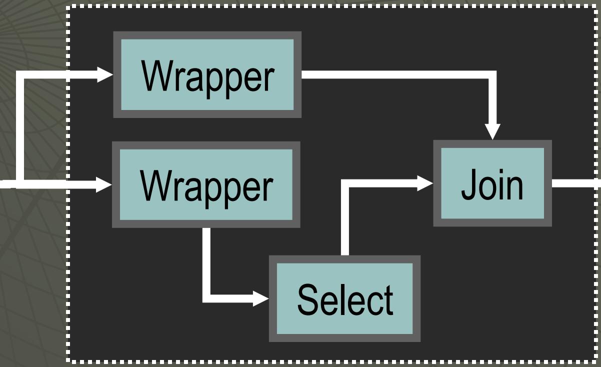
Streaming Dataflow

- Plans consist of a network of operators
  - Examples: **Wrapper**, **Select**, etc.
  - Operators produce and consume data
  - Operators "fire" upon any input data
- Data passed as tuples of a relation

Input relation

City	State	Max Price
Santa Monica	CA	200000

Plan

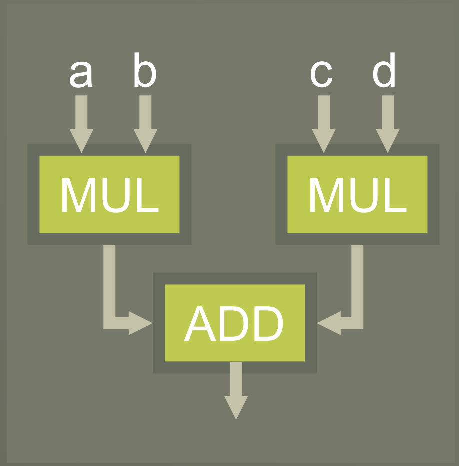


Output relation

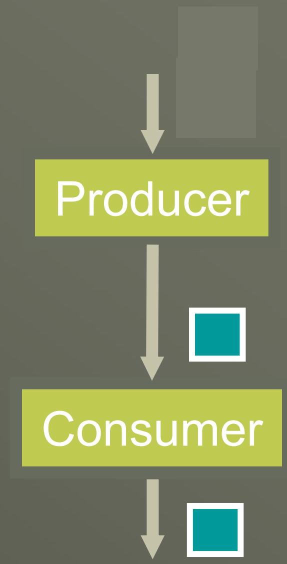
Address
100 Main St., Santa Monica, 90292
520 4th St. Santa Monica, 90292
2 Ocean Blvd, Venice, 90292

Parallelism in Streaming Dataflow

- **<u>Dataflow</u>**
  - Operations scheduled by data availability
    - Independent operations execute in parallel
    - **Maximizes horizontal parallelism**
  - <u>Example</u>: computing **(a*b) + (c*d)**



- **<u>Streaming</u>**
  - Operations emit data as soon as possible
    - Independent data processed in parallel
    - **Maximizes vertical parallelism**



CarInfo Agent

- Agent for recommending used cars:
  - Combine information from
    - Prices of used cars
    - Safety ratings
    - Reviews
  - Example:
    - 2002 Midsize coupe/hatchback
    - $4K-$12K,
    - No Oldsmobiles

The CarInfo agent

1. **Locate cars that meet criteria**
   - Edmunds.com

The CarInfo agent

1. Locate cars that meet criteria
   - Edmunds.com

2. Filter out Oldsmobiles

The CarInfo agent

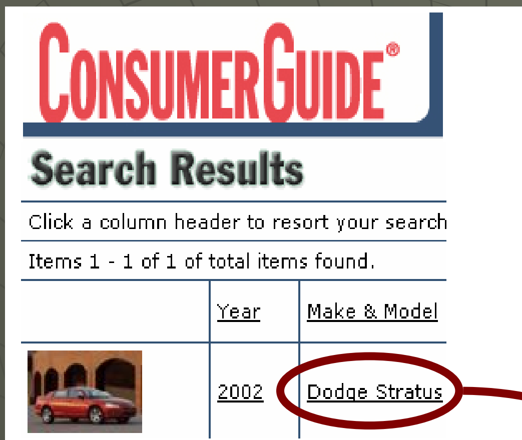
1. **Locate cars that meet criteria**
   - Edmunds.com
2. **Filter out Oldsmobiles**
3. **Gather safety reviews for each**
   - NHSTA.gov

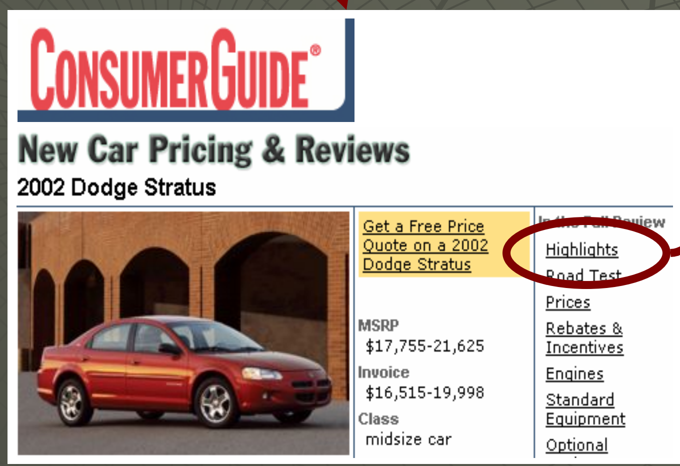
The CarInfo agent

1. **Locate cars that meet criteria**
   - Edmunds.com
2. **Filter out Oldsmobiles**
3. **Gather safety reviews for each**
   - NHSTA.gov
4. **Gather detailed reviews of each**
   - ConsumerGuide.com

ConsumerGuide Navigation

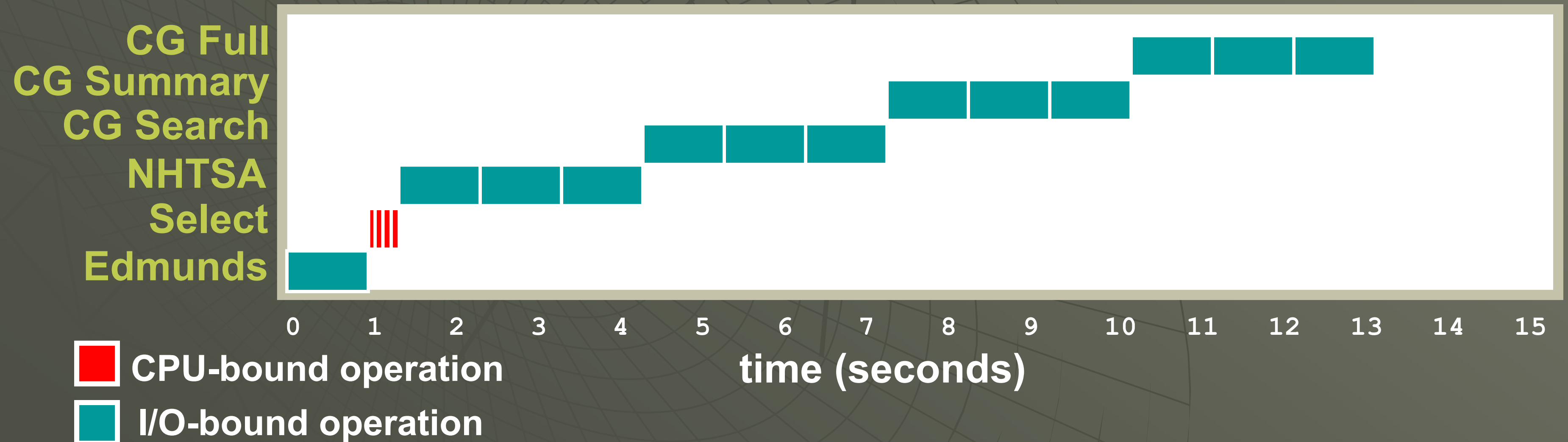
- Requires navigating through multiple pages



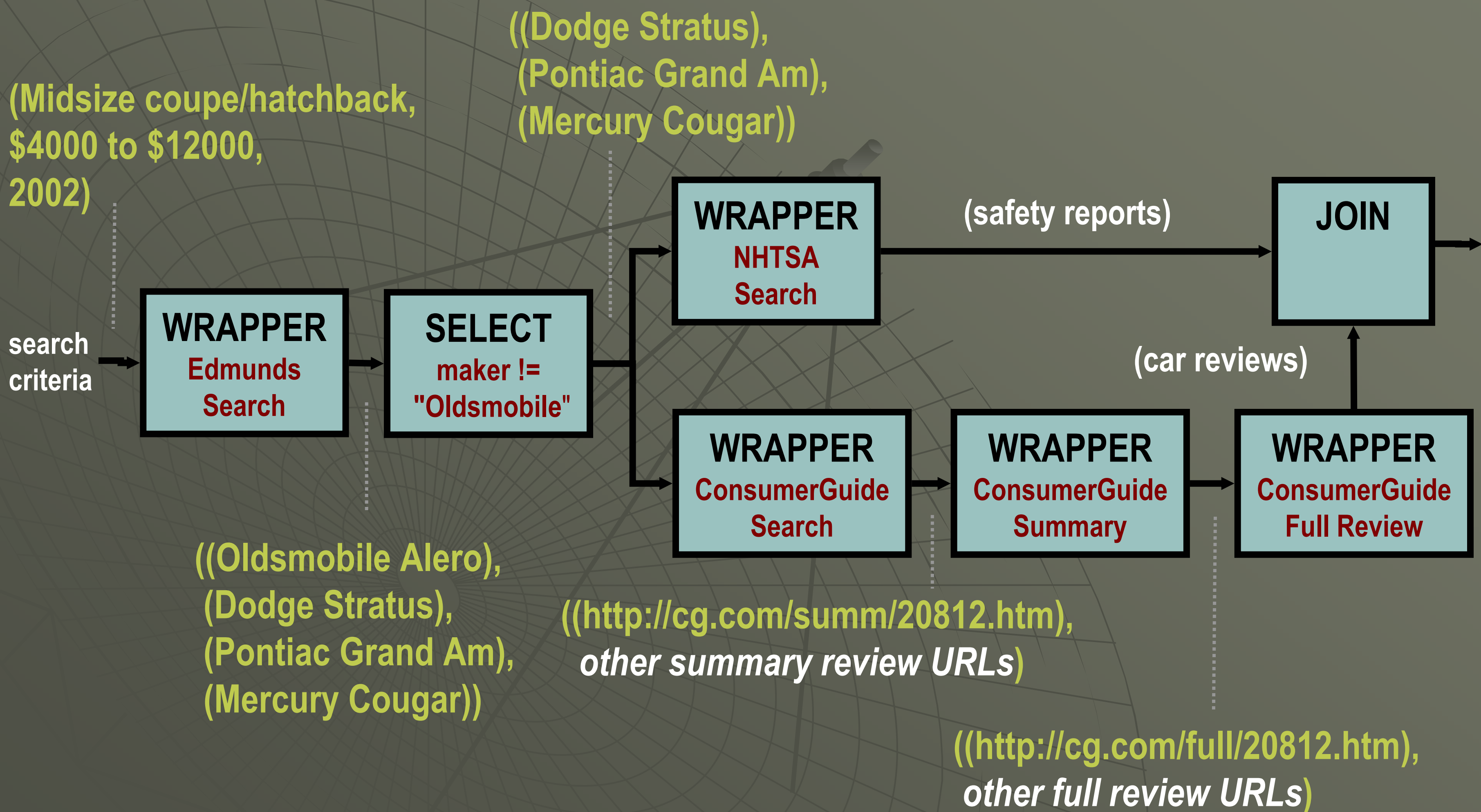


Agent Execution Performance

- Standard von Neumann model
  - Execute one operation at a time
  - Each operation processes all of its input before output is used for next operation
  - **Assume**: 1000ms per I/O op, 100ms per CPU op
- Execution time = 13.4 sec



Dataflow-style CarInfo agent plan



Streaming Dataflow Performance

- Improved, but plan remains I/O-bound (76%)
- Main problem: remote source latencies
  - Meanwhile, local resources are wasted
- Complicating factor: binding constraints
  - Remote queries dependent on other remote queries

- Question: How can execution be more efficient?

Outline

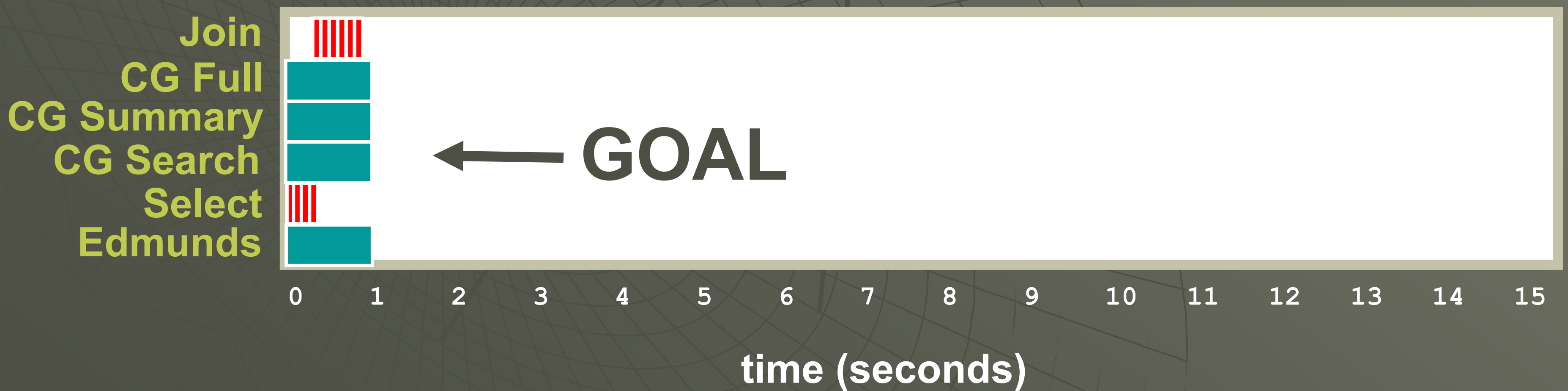
1. Motivating Application: The Electric Elves
2. Efficiently executing agent plans
3. Speculative plan execution
4. Value prediction for speculative execution
5. Related Work
6. Conclusions

Speculative Execution
[Barish & Knoblock '02, '03]

- Basic idea
  - Exploit idle resources to execute future instructions in advance of when they are normally issued
- Challenges
  - How to augment plans for speculation
  - How to ensure correctness and fairness
  - How to decide what to speculate on

Speculative plan execution

- Execute operators ahead of schedule
  - Predict data based on past execution
- Allows greater degree of parallelism
  - Solves the problem caused by binding constraints
- Can lead to speedups > streaming dataflow



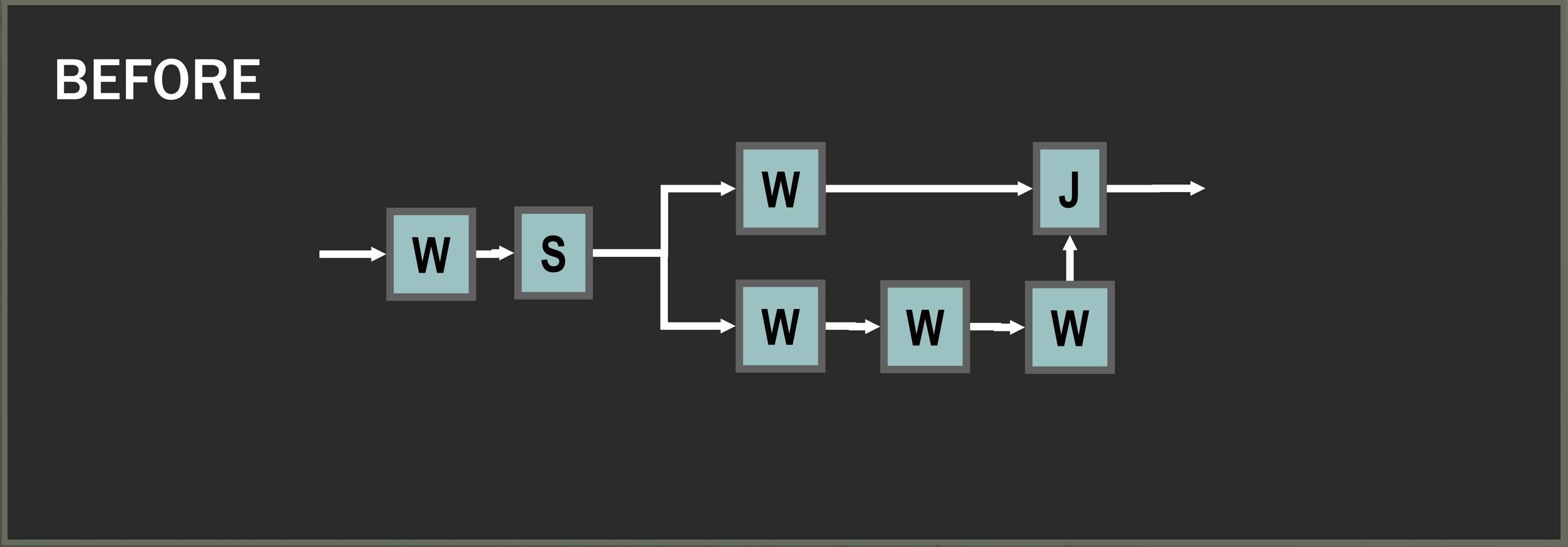
How to speculate?

- General problem
  - Means for issuing and confirming predictions
- Two new operators
  - **Speculate**: Makes predictions based on "hints"

  - **Confirm**: Prevents errant results from exiting plan

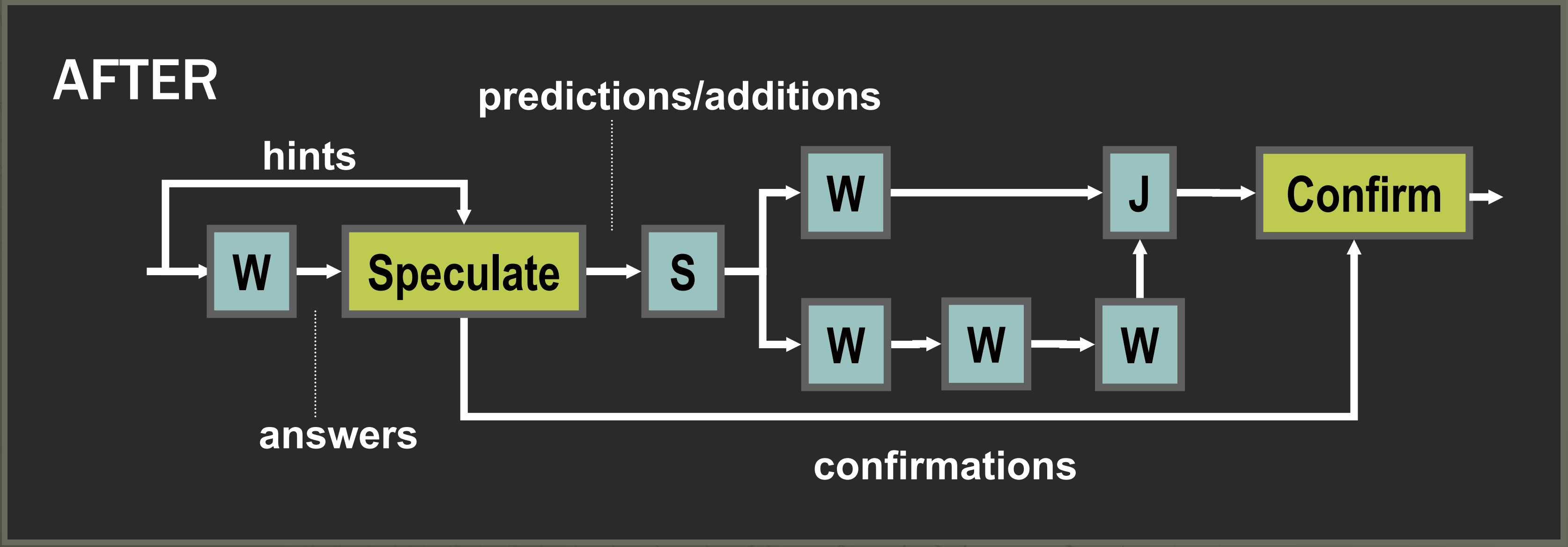
How to speculate?

- Example: CarInfo
  - Predict cars based on search criteria
  - Makes practical sense:
    - Same criteria yields same cars

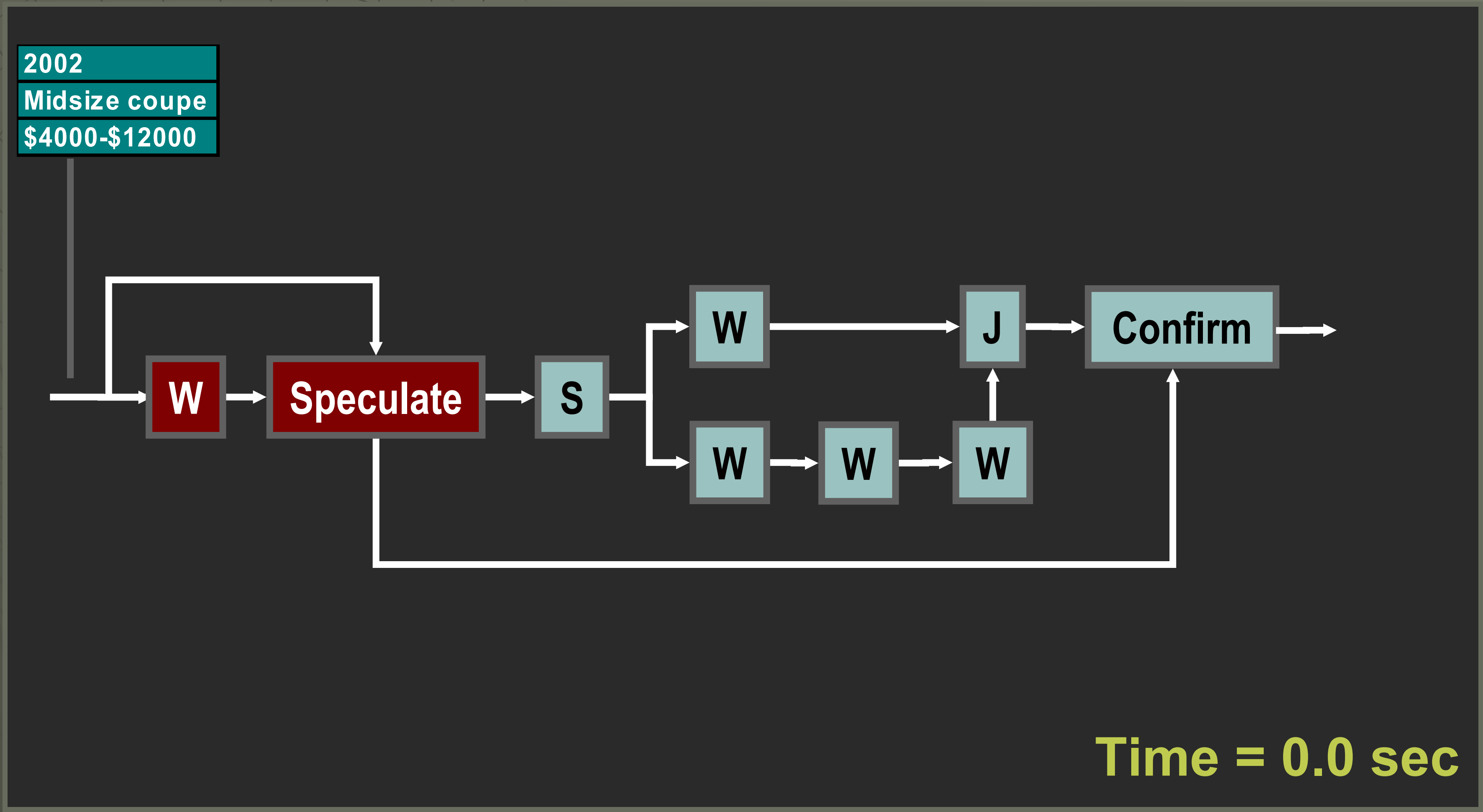


How to speculate?

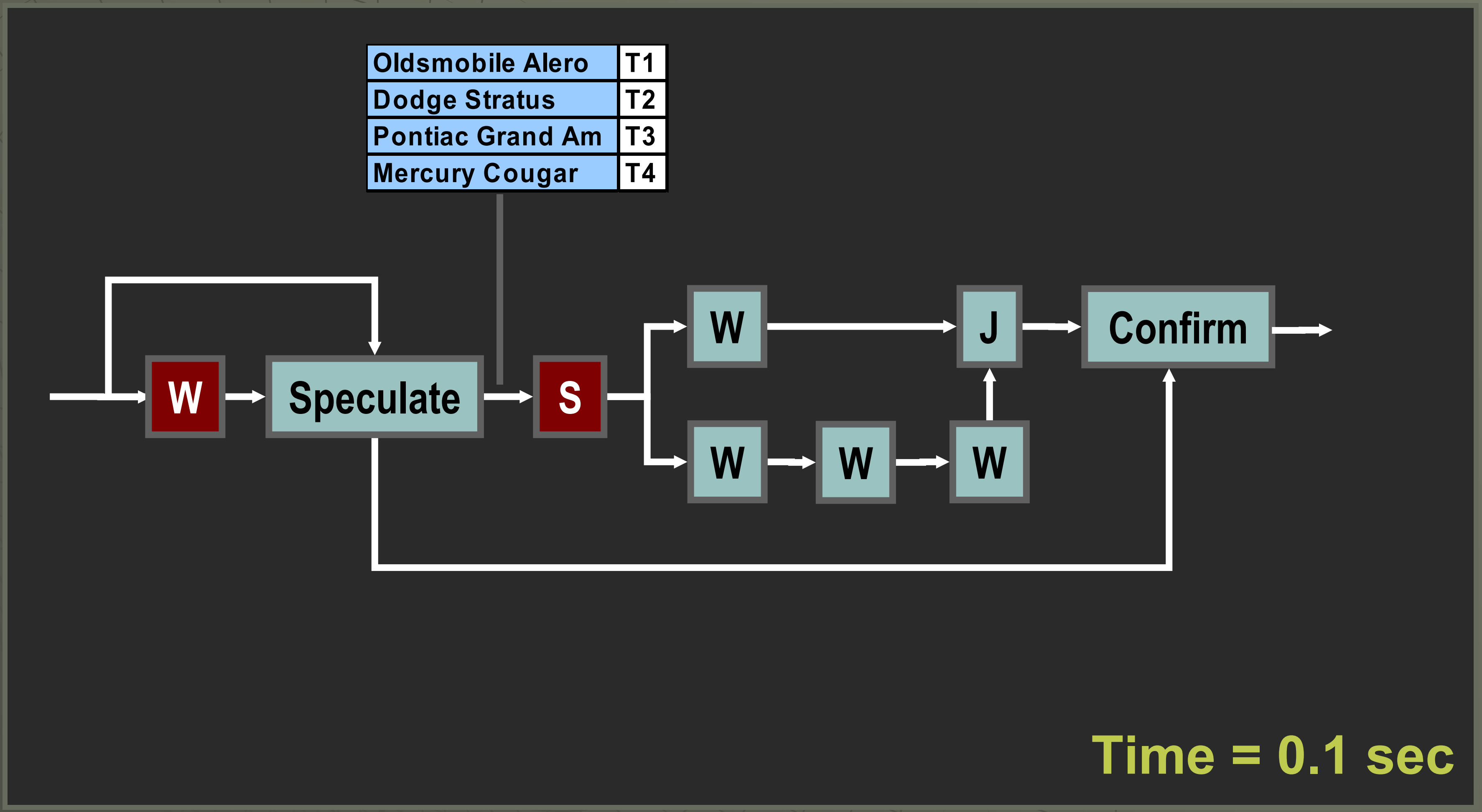
- Example: CarInfo
  - Predict cars based on search criteria
  - Makes practical sense:
    - Same criteria yields same cars



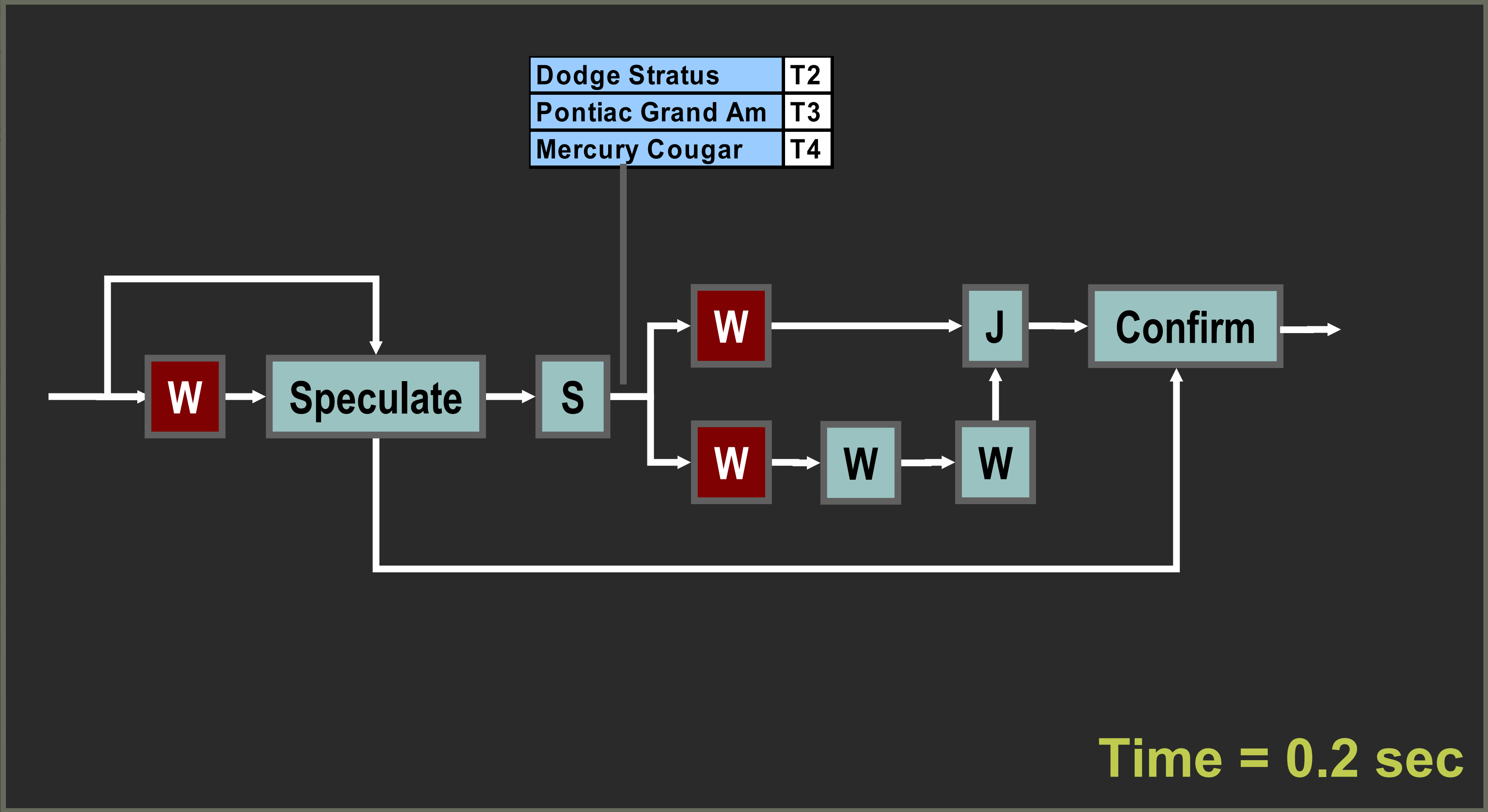
Detailed example



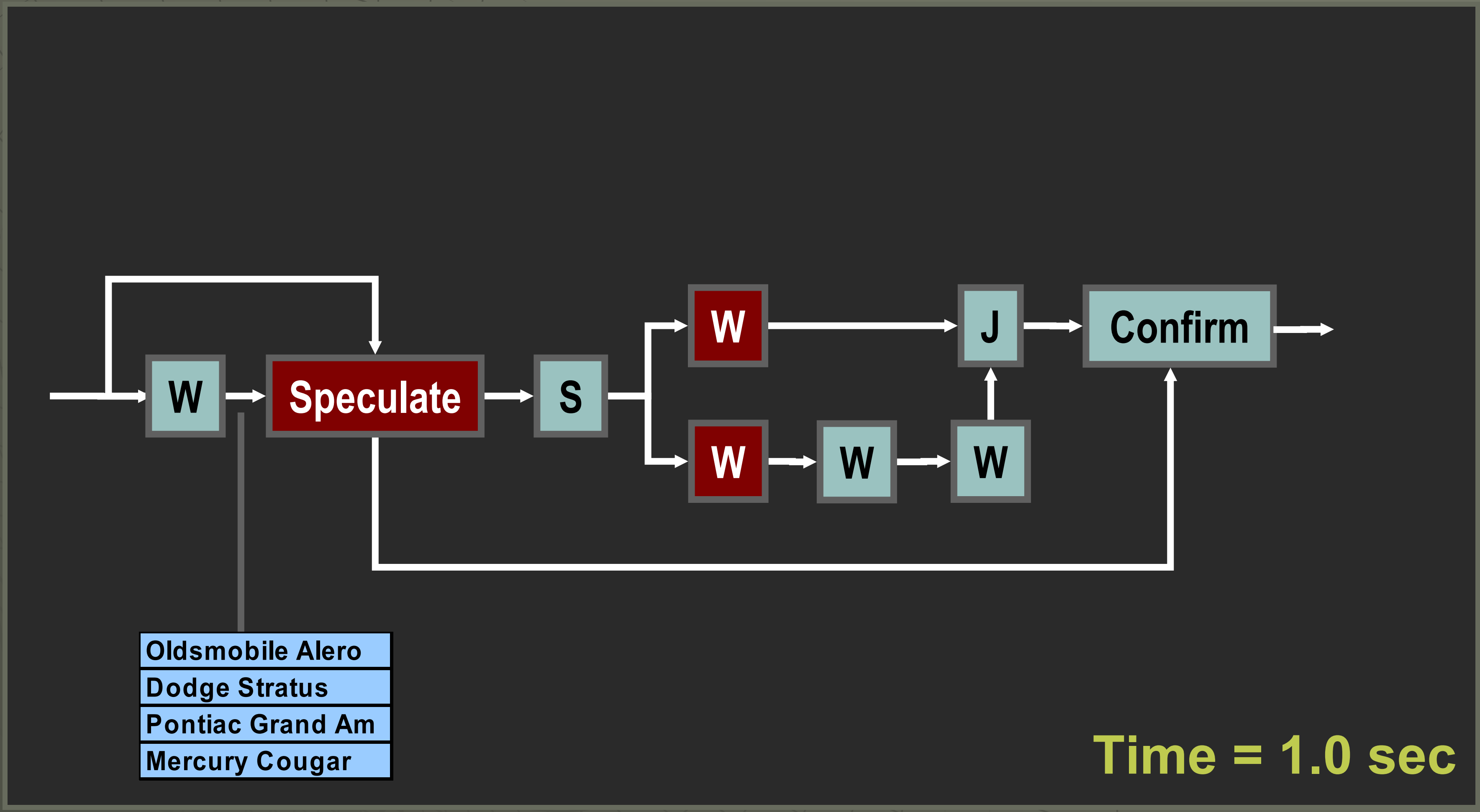
Issuing predictions



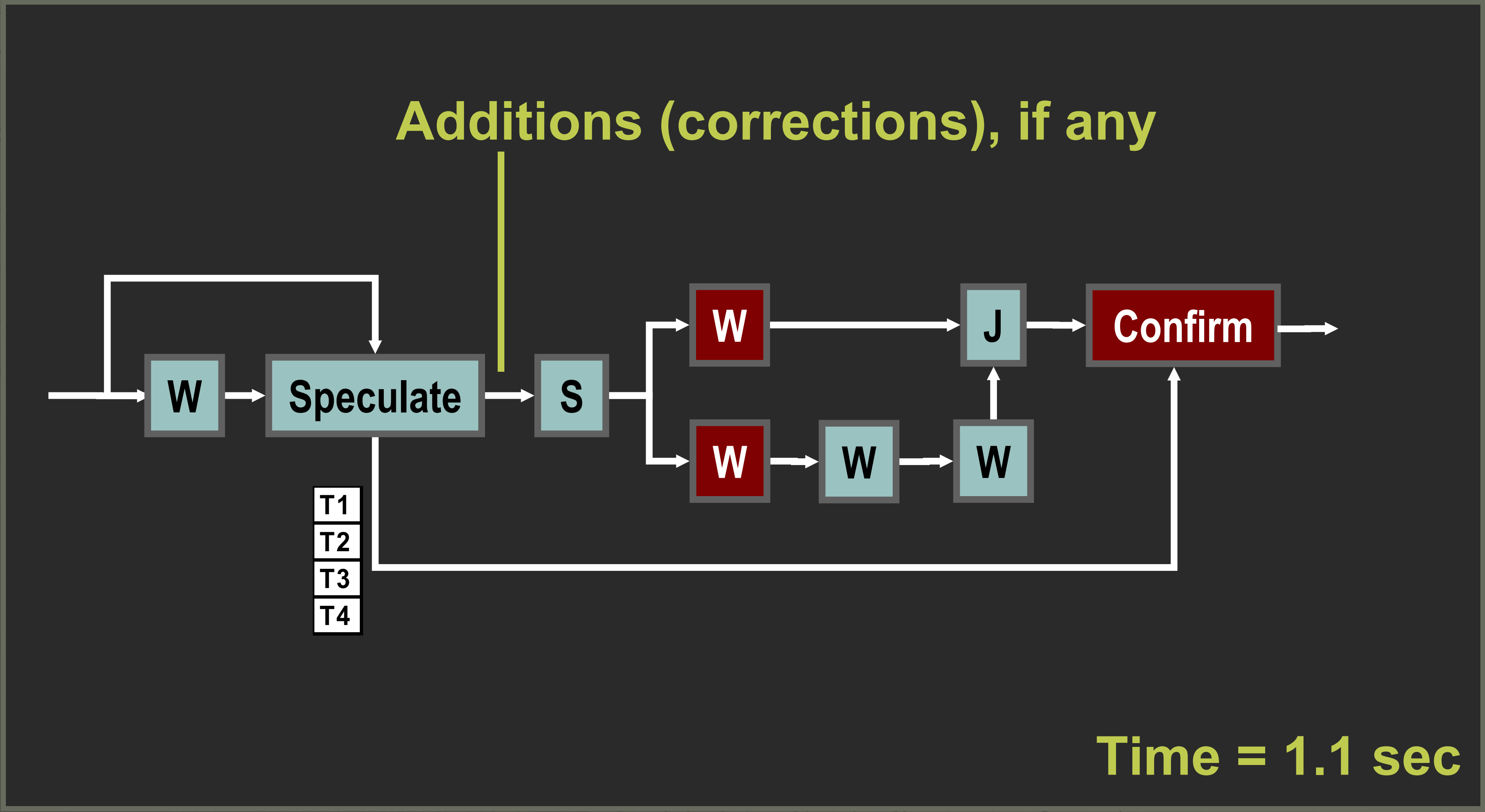
Speculative parallelism



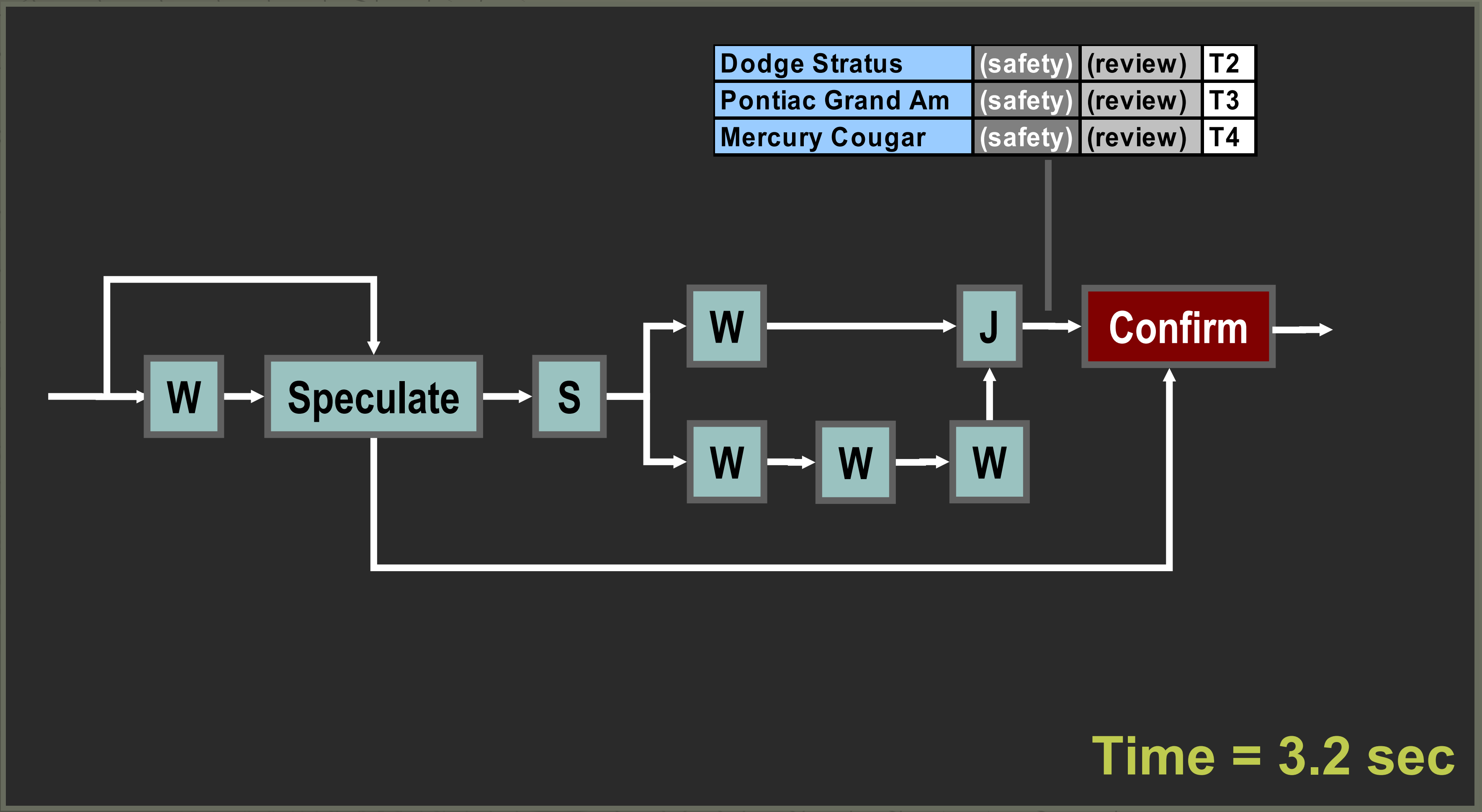
Answers to hints



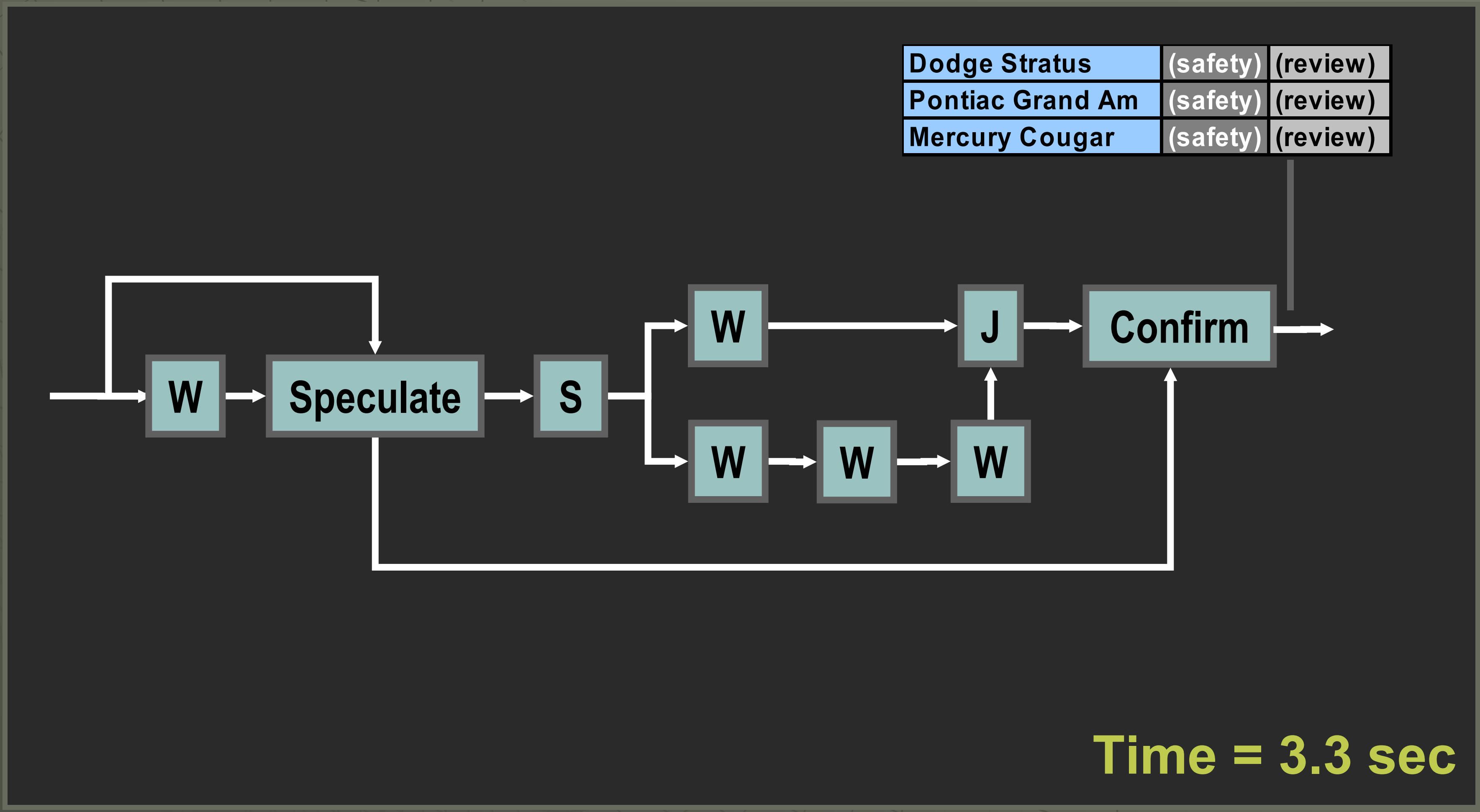
Continued processing



Generation of final results



Confirmation of results



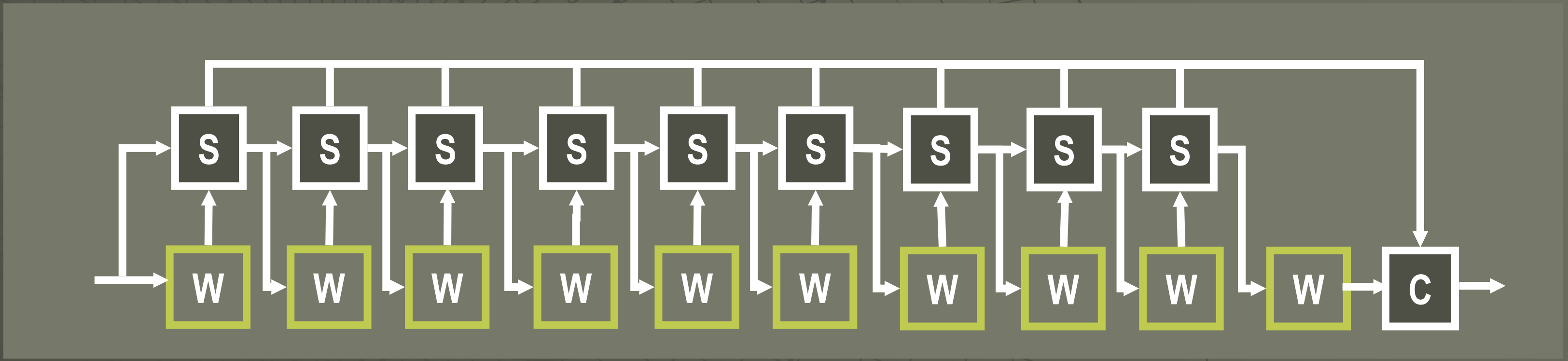
Safety and fairness

- Safety
  - **Confirm** blocks predictions (and results of) from exiting plan before verification

- Fairness
  - **CPU**
    - Speculative operations use "speculative threads"
      - Lower priority threads
  - **Memory and bandwidth**
    - Speculative operations allocate "speculative resources"
      - Drawn from "speculative pool" of memory / objects

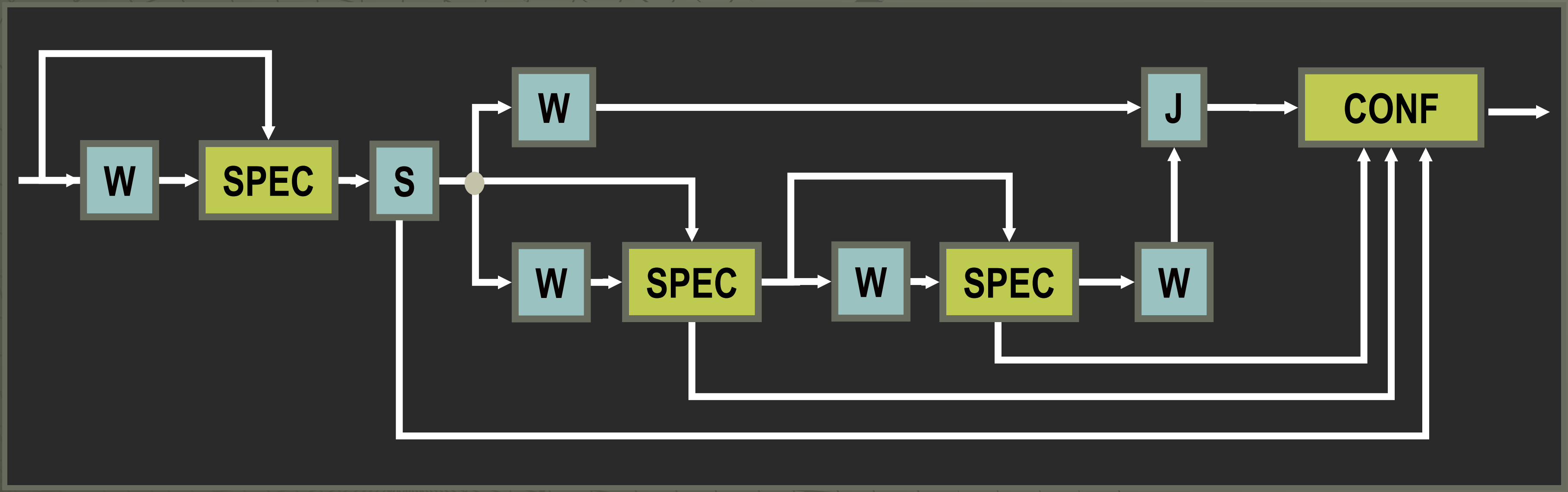
Getting better speedups

- Cascading speculation
  - Single speculation allows a max speedup of 2
    - **Time spent either speculating or confirming**
  - Cascading speculation allows **arbitrary speedups**
    - **Up to the length of the longest plan flow**



Cascading Speculation

- Use predicted cars to speculate about the ConsumerGuide summary and full URLs



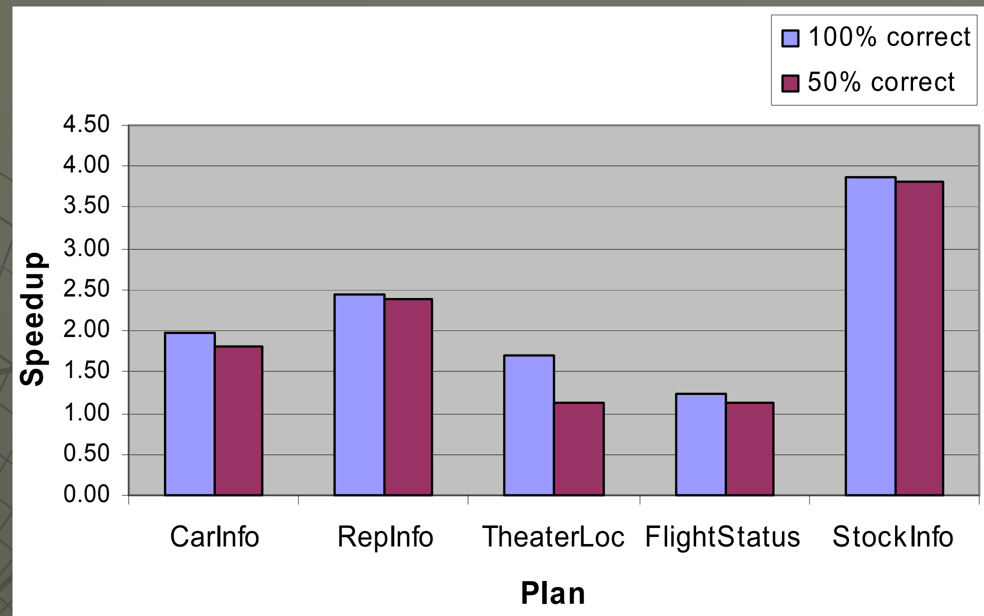
- Optimistic performance
  - Execution time: **max** {1.2, 1.4, 1.5, 1.6} = **1.6 sec**
  - Speedup over streaming dataflow: (4.2/1.6) = **2.63**

Automatic plan transformation

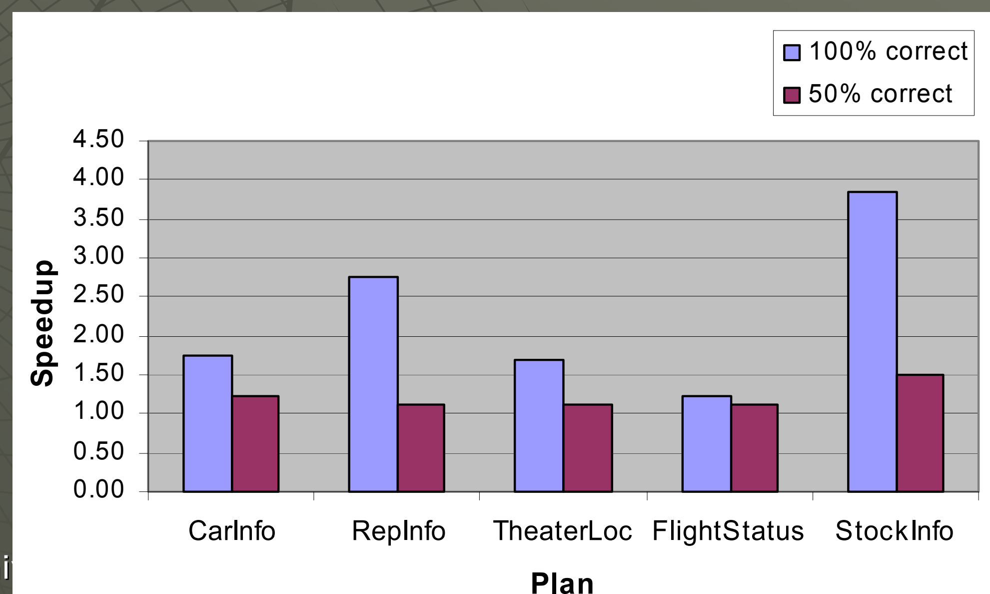
- Agent plans are automatically modified for speculative execution
  - Successive runs of the plan benefit
    - Even with different input data
- Optimize only the most expensive path (**MEP**)
- Algorithm
  1. **Find MEP**
  2. **Find best candidate speculative plan transformation**
  3. **IF no candidate found, THEN exit**
  4. **Transform plan accordingly**
  5. **REPEAT** **(anytime property)**

Speculation Results

- Time to first tuple:



- Time to last tuple:



Outline

1. Motivating Application: The Electric Elves
2. Efficiently executing agent plans
3. Speculative plan execution
4. Value prediction for speculative execution
5. Related Work
6. Conclusions

Value prediction

- Better value prediction = better speedups
- Prediction capability

Category	Hint	Prediction
A	Previously seen	Previously seen
B	Never seen	Previously seen
C	Never seen	Never seen

- Examples:

Edmunds car list from **search criteria**

5K-12K ?

H → **2002 Midsize coupe 4K-12K**

P → **Olds Alero, Dodge Stratus, Pontiac Grand Am, Mercury Cougar**

ConsumerGuide full review URL from **summary URL**

http://cg.com/summary/20812.htm

http://cg.com/full/20812.htm

http://cg.com/summary/12345.htm ?

Learning for Speculative Execution

- Caching
  - Associate a hint with a predicted value
    - 2002 Midsize coupe 4K-12K
      → Olds Alero, Dodge Stratus, Pontiac Grand Am, Mercury Cougar
- Classification
  - Use features of a hint to predict value
    - **EXAMPLE:** Predicting car list from Edmunds

Year	Type	Min	Max	Car list
2002	Midsize	8000	15000	(Oldmobile Alero, Dodge Stratus)
2002	Midsize	7500	14500	(Oldmobile Alero, Dodge Stratus)
2002	SUV	14000	20000	(Nissan Pathfinder, Ford Explorer)
2001	Midsize	11000	18000	(Honda Accord, Toyota Camry)
2002	SUV	18000	22000	(Nissan Pathfinder, Ford Explorer)

← **Cache**

Decision list →

type = SUV : (Nissan Pathfinder, Ford Explorer)
type = Midsize :
:...min <= 10000 : (Olds Alero, Dodge Stratus)
min > 10000 : (Honda Accord, Toyota Camry)

Learning for Speculative Execution

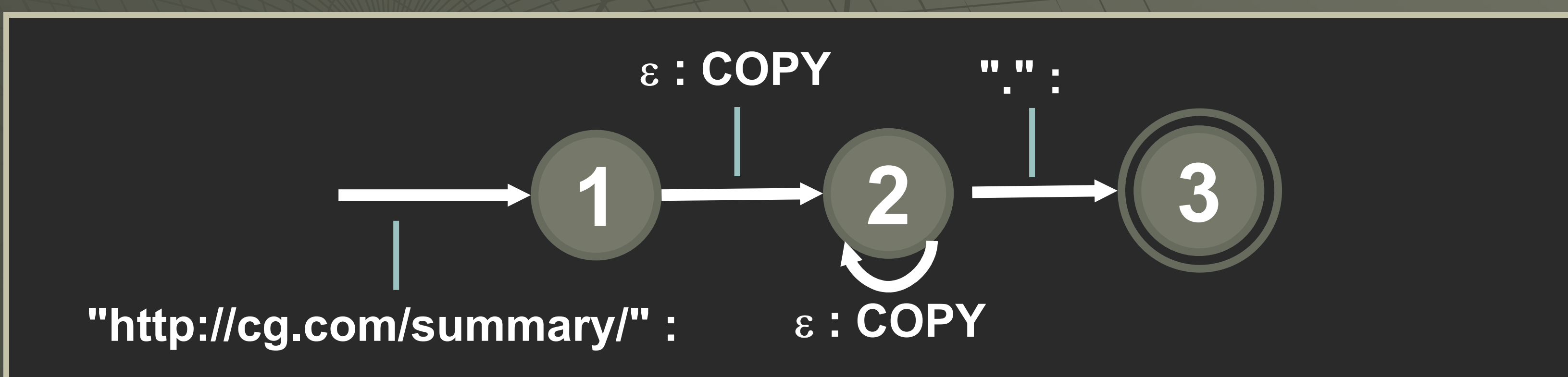
- Transduction
  - Transducers are FSM that translate hints into predictions

http://cg.com/summary/20812.htm

http://cg.com/full/20812.htm

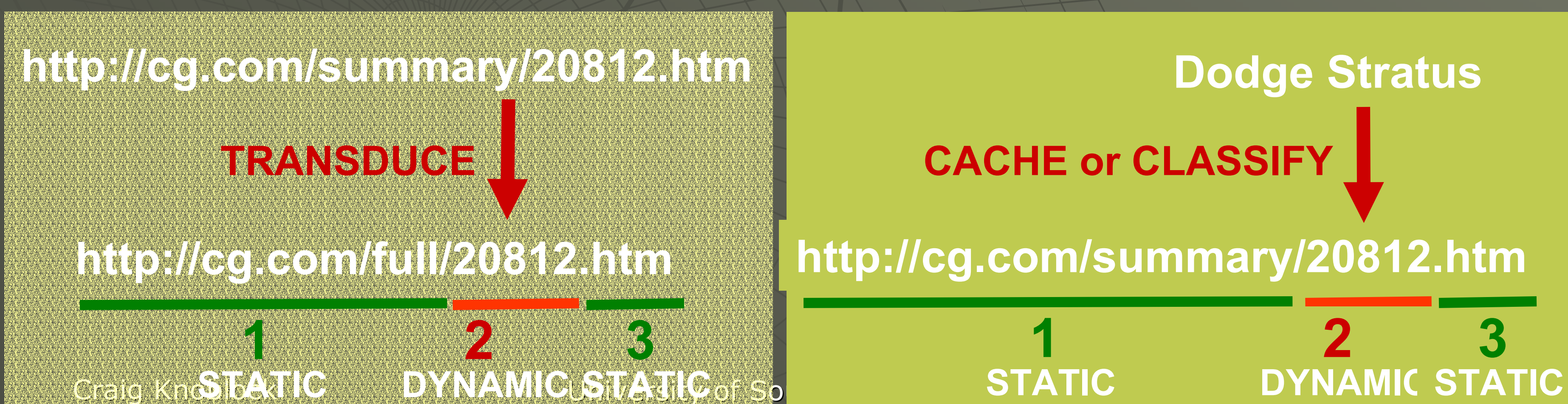
To create full review URL:

1. Insert "http://cg.com/full/"
2. Extract & insert the dynamic part of the summary URL (e.g., 20812)
3. Insert ".htm"



Unifying Algorithm: Learn Value Transducers

- Combines different learning methods
- Identify predicted value "templates"
  - Alternating seq of STATIC/DYNAMIC elements
- Value transducers built from templates
  - State transitions (arcs) = high-level operations:
    - **INSERT, CACHE, CLASSIFY, TRANSDUCE**



Learning value transducers

- Identify STATIC/DYNAMIC template
  - Find LCS for the set of predicted values, using technique based on (Hirschberg 1975)
- For each STATIC element,
  - Construct **INSERT** arc to next automata state
- For each DYNAMIC element,
  - Construct **TRANSDUCE**, **CLASSIFY**, or **CACHE** arc to next automata state
    - Prefer TRANSDUCE and CLASSIFY because
      - Better predictive capability on average
      - Better space efficiency on average

Detailed example: CarInfo URLs

HINTS:

http://cg.com/summary/20812.htm

http://cg.com/summary/12345.htm

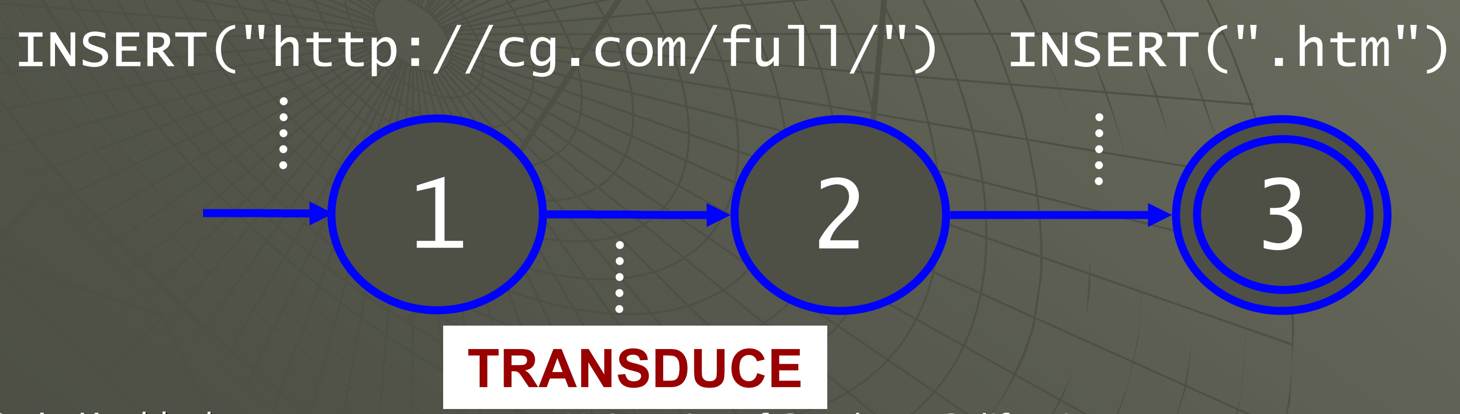
ANSWERS:

http://cg.com/full/20812.htm

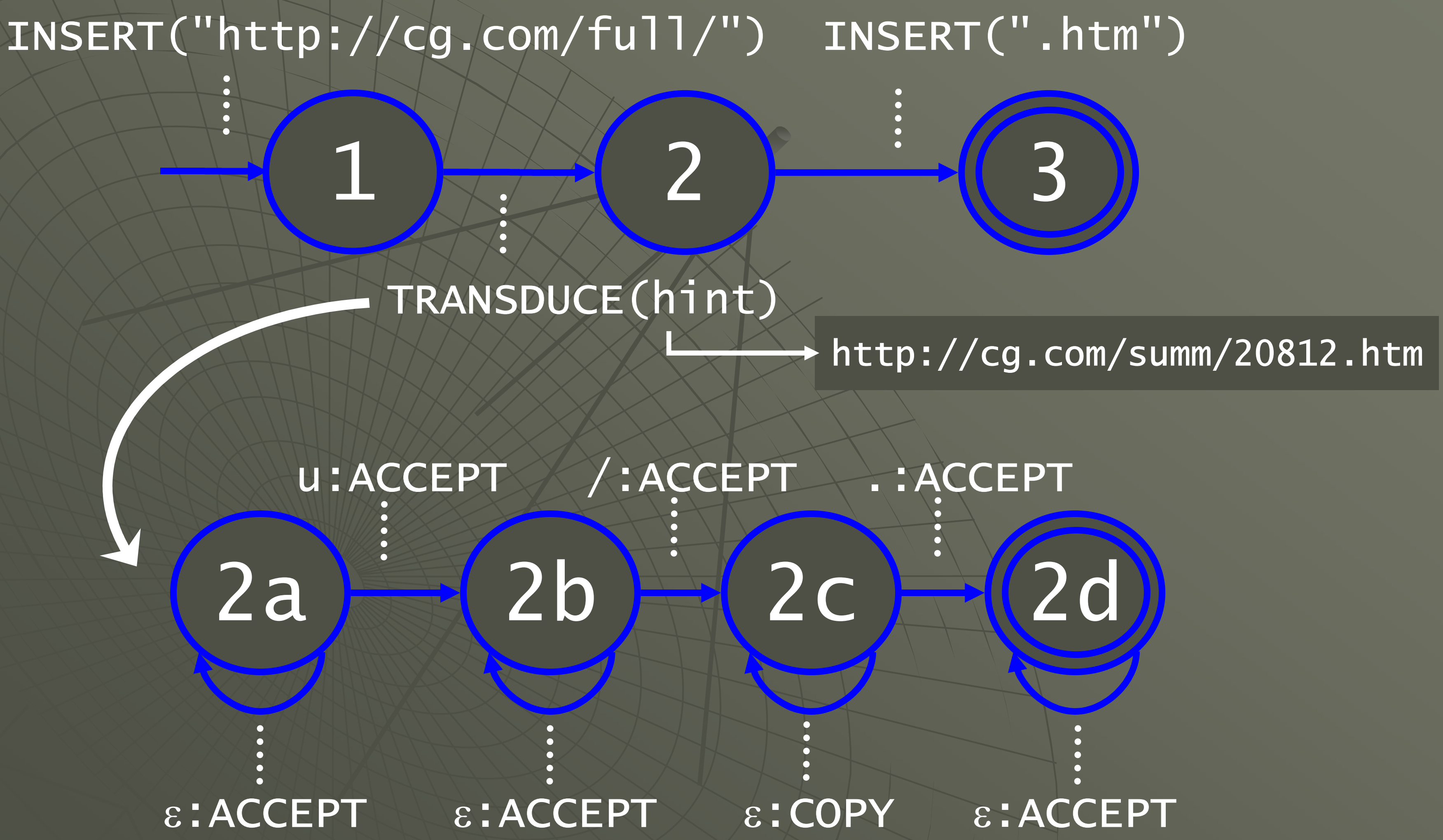
http://cg.com/full/12345.htm

TEMPLATE

http://cg.com/full/[DYNAMIC].htm

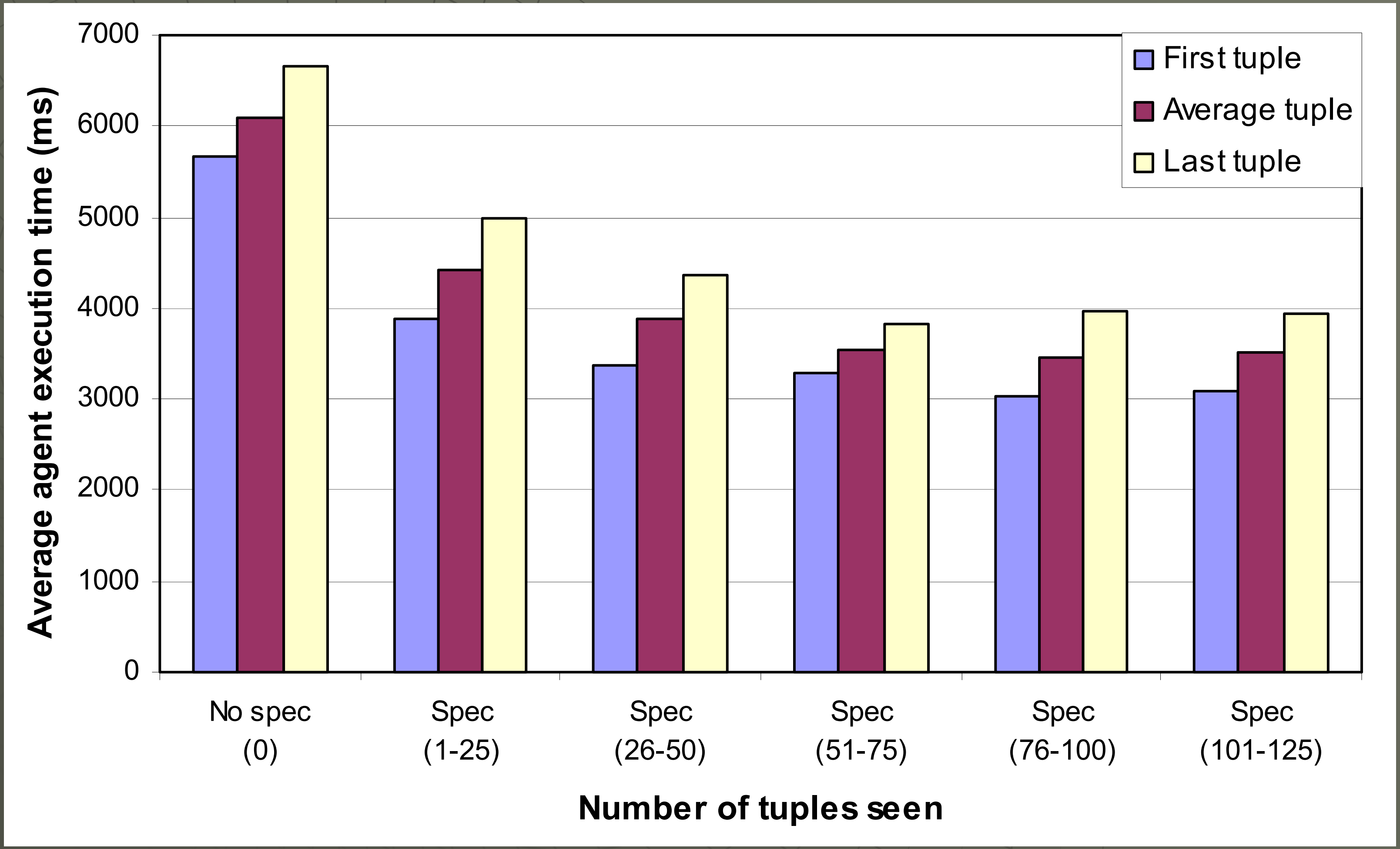


Example Value Transducer



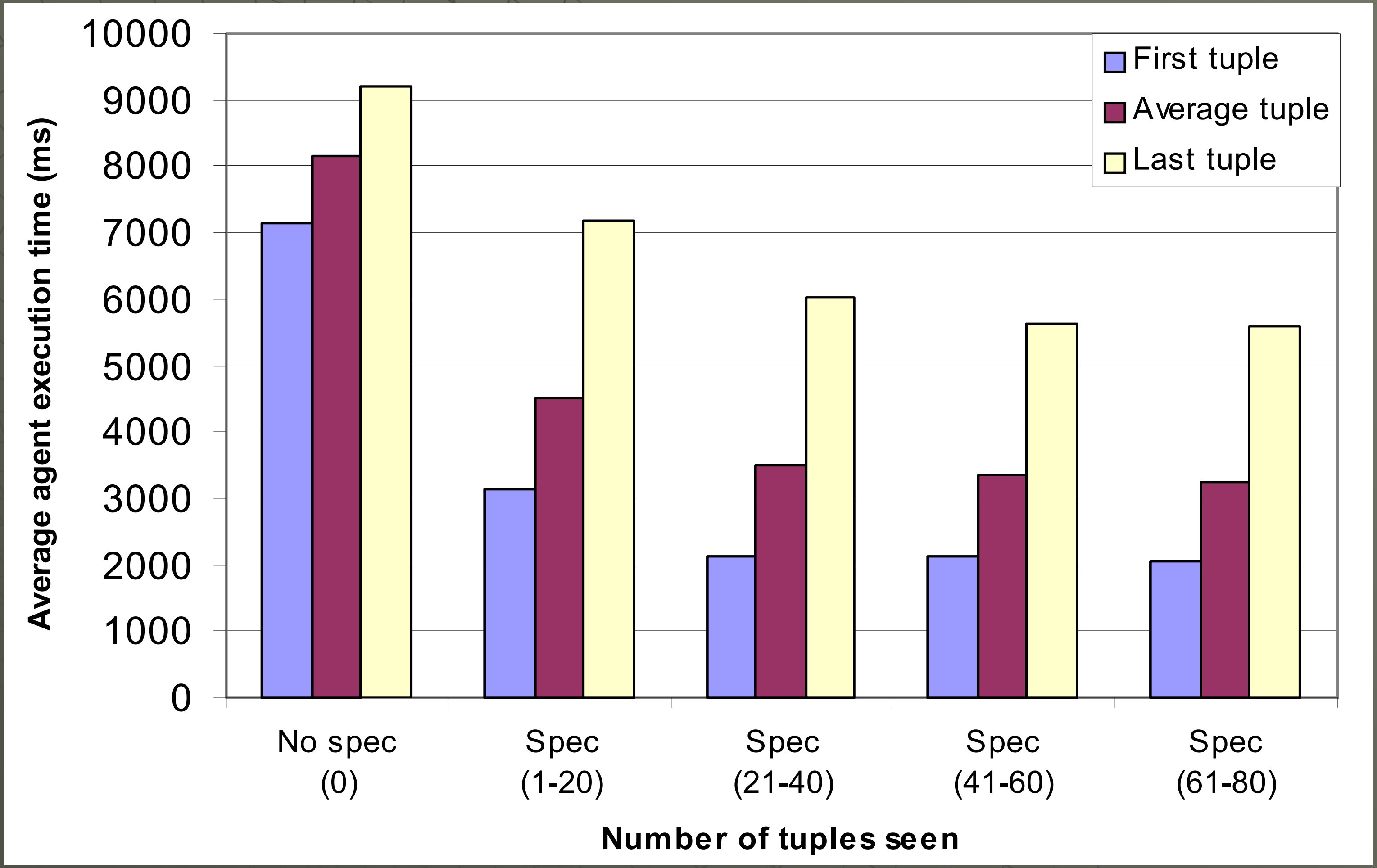
Effect on spec exec performance

- CarInfo



Effect on spec exec performance

- RepInfo



Outline

1. Motivating Application: The Electric Elves
2. Efficiently executing agent plans
3. Speculative plan execution
4. Value prediction for speculative execution
5. Related Work
6. Conclusions

Related Work

- Speculative execution
  - **Approximate & partial query results**
    - **[Hellerstein et al. 1997] [Shanmugasundaram et al. 2000] [Raman and Hellerstein 2001]**
  - **Executing anticipated actions in advance**
    - Continual computation **[Horvitz 2001],** time-critical decision making **[Greenwald and Dean 1994]**
  - **Other types of speculative execution**
    - File system prefetching **[Chang and Gibson 1999],** control speculation in workflow processing **[Hull et al. 2000]**
  - **Network prefetching**

Related Work

- Learning value predictors
  - Predicting commands
    - **Command line prediction** **[Davison and Hirsh 1998, 2001]**
    - **Assisted browsing** **[Lieberman 1995] [Joachims et al. 1997]**
  - Value prediction as speedup learning
    - **[Fikes et al. 1972], [Mitchell 1983], [Minton 1988]**
  - Transducer learning
    - **Provably correct transducers** **[Oncina et al. 1993]**
      - Issues: Requires many examples, generalization capability differs
    - **Transducers for data extraction** **[Hsu and Chang 1999]**
  - URL prediction
    - **[Zukerman et al. 1999], [Su et al. 2000]**

Outline

1. Motivating Application: The Electric Elves
2. Efficiently executing agent plans
3. Speculative plan execution
4. Value prediction for speculative execution
5. Related Work
6. Conclusions

Summary

- An approach to speculative execution of information agent plans
  - Can yield arbitrary speedups
  - Safe, fair
- Value prediction approach that combines caching, classification, and transduction
  - More accurate & space efficient than strictly caching

Conclusions

- Speculative execution is a form of speed-up learning
- Two very large search spaces:
  - Plan transformations for speculative execution
  - Value prediction for each speculate operator
- Both of these are potential opportunities for CBR in information gathering
  - Could learn finer-grained plan transformations that depend on the request
  - Could learn more sophisticated value prediction strategies (e.g., speculating on multiple inputs)
- Finding the right speculative plan and value predictions can provide significant speedups!

More Information

- My home page: http://www.isi.edu/~knoblock
- Papers:
  - Theseus execution system: JAIR'05
  - Speculative execution: ICAPS'02
  - Value prediction: IJCAI'03
  - Electric Elves: IAAI'01, IAAI'02
- Thesis:
  - Greg Barish, 2004, USC

The End