

Politecnico di Milano
Dipartimento di Elettronica e Informazione

&



A First Step Towards Stream Reasoning

Presenter:

Emanuele Della Valle

emanuele.dellavalle@polimi.it

<http://emanueledellavalle.org>

Authors:

Emanuele Della Valle, Stefano Ceri, Davide F. Barbieri,
Daniele Braga and Alessandro Campi



Agenda



- Introduction
 - Motivation
 - Urban Computing
- Problem definition
 - Knowledge and data can change over time
 - Data Stream Management Systems
 - Is Stream Reasoning possible?
- A Conceptual Architecture for Stream Reasoning
 - The LarKC Project
 - A conceptual architecture
 - RDF streams
 - C-SPARQL
- Two approaches to stream reasoning
 - Evolutionary
 - Revolutionary
- Conclusions
 - Research agenda - medium term
 - Research agenda - long term

Introduction



- While **reasoners are** year after year **scaling up** in the classical, time invariant domain of ontological knowledge, **reasoning upon rapidly changing information has been neglected or forgotten.**
- **Data streams** are *unbounded sequences of time-varying data elements*;
 - **They occur in a variety of modern applications**, such as network monitoring, traffic engineering, sensor networks, RFID tags applications, telecom call records, financial applications, Web logs, click-streams, etc.
 - processing of **data streams has been largely investigated** and specialized Stream Database Management Systems exist.
- The combination of reasoning techniques with data streams gives rise to **Stream Reasoning, an unexplored, yet high impact, research area.**

Motivation



- **To understand the potential impact of Stream Reasoning, we can consider the emblematic case of Urban Computing**

- The integration of computing, sensing, and actuation technologies into everyday urban settings and lifestyles.

[source [IEEE Pervasive Computing, July-September 2007 \(Vol. 6, No. 3\)](#)]

- **Pervasive computing** has been applied
 - either in relatively **homogeneous rural areas**, where researchers have added sensors in places such as forests, vineyards, and glaciers
 - or, on the other hand, **in small-scale**, well-defined patches of the built environment **such as smart houses or rooms**.
- **Urban settings**
 - include, for example, *streets, squares, pubs, shops, buses, and cafés* - any space in **the semipublic realms** of our towns and cities.
 - **are challenging** for experimentation and deployment, **and they remain little explored**

Availability of Data



- **Some years ago**, due to the lack of data, Urban Computing would have looked like a **Sci-Fi idea**.
- **Nowadays**, a large amount of the required **information** can be made **available** on the Internet at almost no cost:
 - maps with the commercial activities and meeting places,
 - events scheduled in the city and their locations,
 - average speed in highways, but also normal streets
 - positions and speed of public transportation vehicles
 - parking availabilities in specific parking areas,
 - and so on.
- We are running a survey (please contribute), see
 - <http://wiki.larkc.eu/UrbanComputing/ShowUsABetterWay>
 - <http://wiki.larkc.eu/UrbanComputing/OtherDataSources>

A challenge for Stream Reasoning



- **Looking for parking lots** in large cities may cost up to 40% of the daily fuel consumption. Problems dramatically increase when big events, involving lots of people, take place
- **A typical Urban Computing problem** is
 - to **help citizens** willing to participate to such events **in finding a parking lot** and reaching the event locations in time,
 - **while globally limiting** the occurrences of **traffic congestions**.
- Current technologies are not up to this challenge, because **it requires**
 - **combining**
 - a huge amount of **static knowledge** about the city
 - with an even **larger set of data streams**
 - **reasoning** in realtime above the resulting time-varying knowledge.



[source: <http://gizmodo.com/photogallery/trafficsky/1003143552>]

Problem definition



- **Knowledge and data can change over the time.**
 - For instance, in Urban Computing names of streets, landmarks, kind of events, etc. change very slowly, whereas the number of cars that go through a traffic detector in five minutes changes very fast.
- This means that the system must have the **notion of "observation period"**, defined as the period **when the system is subject to querying**.
- Moreover the system, **within a given observation period**, must consider the following **four different types of knowledge and data**:
 - **Invariable knowledge** (a design constrain that we'd like to relax in future)
 - **Invariable data**
 - **Periodically changing data** that change according to a temporal law that can be
 - **Event driven changing data** that are updated as a consequence of some external event.

Invariable knowledge and data

■ Invariable knowledge

- it includes obvious **terminological knowledge**
 - such as an address is made up by a street name, a civic number, a city name and a ZIP code
- less obvious **nomological knowledge** that describes **how the world is expected**
 - **to be**
 - traffic lights are switched off during the night
 - **to evolve**
 - traffic jams appears when important sport events take place

■ Invariable data

- do not change in the observation period, e.g. the names and lengths of the roads.



In the observation period!

Changing data

- **Periodically changing data** change according to a temporal law that can be
 - **Pure periodic law**, e.g. every night at 10pm all Milano overpasses close.
 - **Probabilistic law**, e.g. traffic jam appear in the west side of Milano due to bad weather or when San Siro stadium hosts a soccer match.
- **Event driven changing data** are updated as a consequence of some external event. They can be **further characterized by the mean time between changes**:
 - **Slow**, e.g. roads closed for scheduled works
 - **Medium**, e.g. roads closed for accidents or congestion due to traffic
 - **Fast**, e.g. the intensity of traffic for each street in a city

Google [Advanced Search](#) [Preferences](#)

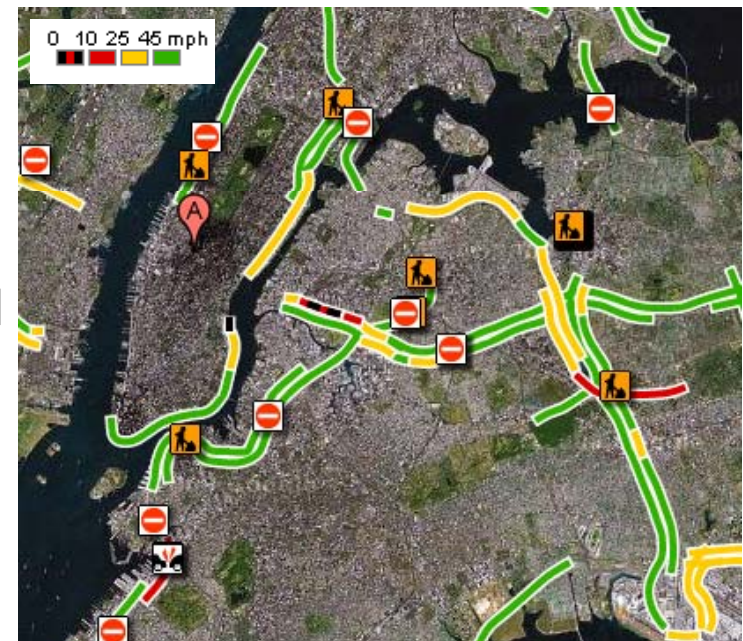
Web

[Inter incontra tifosi: tensione con milanisti e traffico bloccato ...](#) - [[Translate this page](#)]
Inter incontra tifosi: tensione con milanisti e **traffico bloccato** ... Garibaldi / Niguarda - Navigli / Vigentino - **San Siro** / Baggio - Vittoria / Forlanini ...
[www.blogmilano.it/2007/05/25/inter-incontra-tifosi-tensione-con-milanisti-e-traffico-bloccato/-75k](#) - [Cached](#) - [Similar pages](#)

[\[PDF\] Monsignor Angelo Bagnasco nuovo Arcivescovo di Genova](#)
File Format: PDF/Adobe Acrobat - [View as HTML](#)
situazione di alcuni abitanti di corso Magellano: **traffico bloccato** dagli ... **San Siro**, ed in effetti riteniamo che la scelta del Papa, certo fatta ...
[www.stedo.it/settembre_D6/pag4.pdf](#) - [Similar pages](#)

[I concerti a San Siro si faranno](#) - [[Translate this page](#)]
Senza contare poi il **traffico**, che **blocca** buona parte delle strade del circondario sia ... lo show inizia prima
VivimilanoMusica: Vasco Rossi a **San Siro**, ...
[www.02blog.it/post/1584/i-concerti-a-san-siro-si-faranno](#) - 72k - [Cached](#) - [Similar pages](#)

[CLUB METRO - METRO NEWS ITALIA](#) - [[Translate this page](#)]

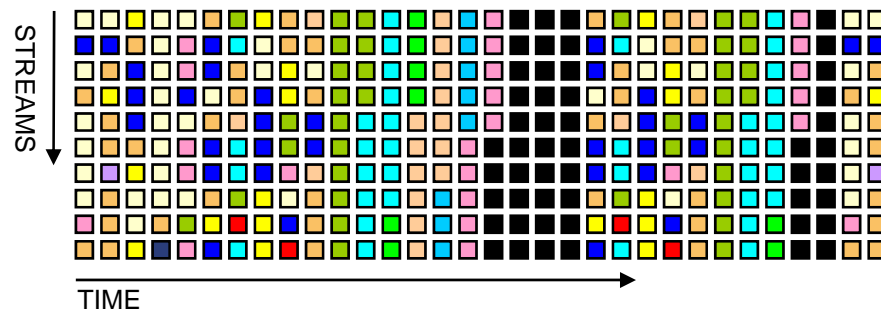


Data Stream Management Systems

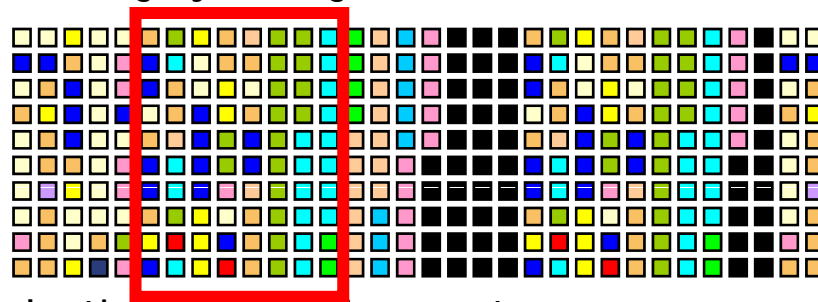


Background: stream database key concepts

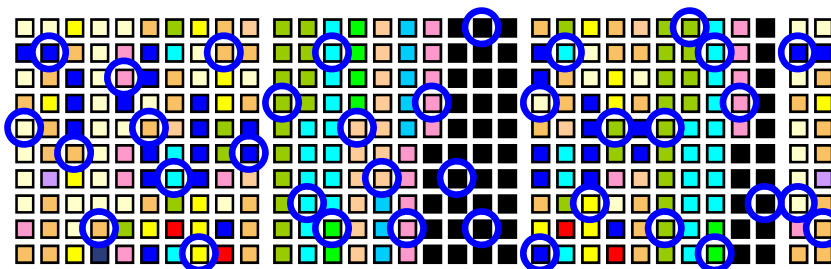
- Streams: continuous instead of one-time semantics



- Selecting by sliding Windows on streams

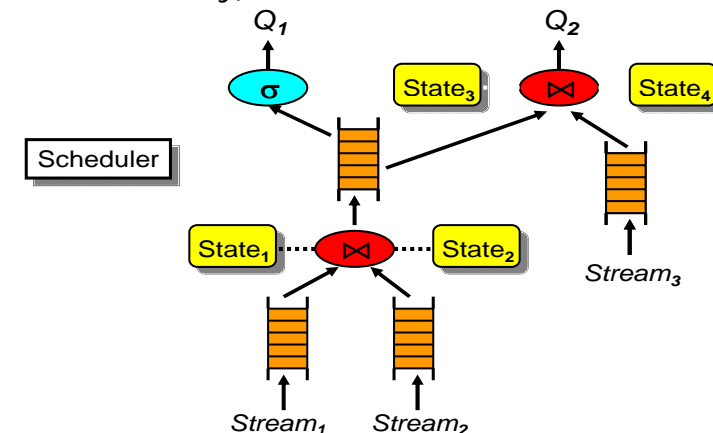


- Selecting by sampling on streams



Query Execution

- When a continuous query is registered, generate a query execution plan
- New plan merged with existing plans
- Plans composed of three main components:
 - Operators
 - Queues (input and inter-operator)
 - State (windows, operators requiring history)



- Global scheduler for plan execution maximizing experience gathered with previous queries.

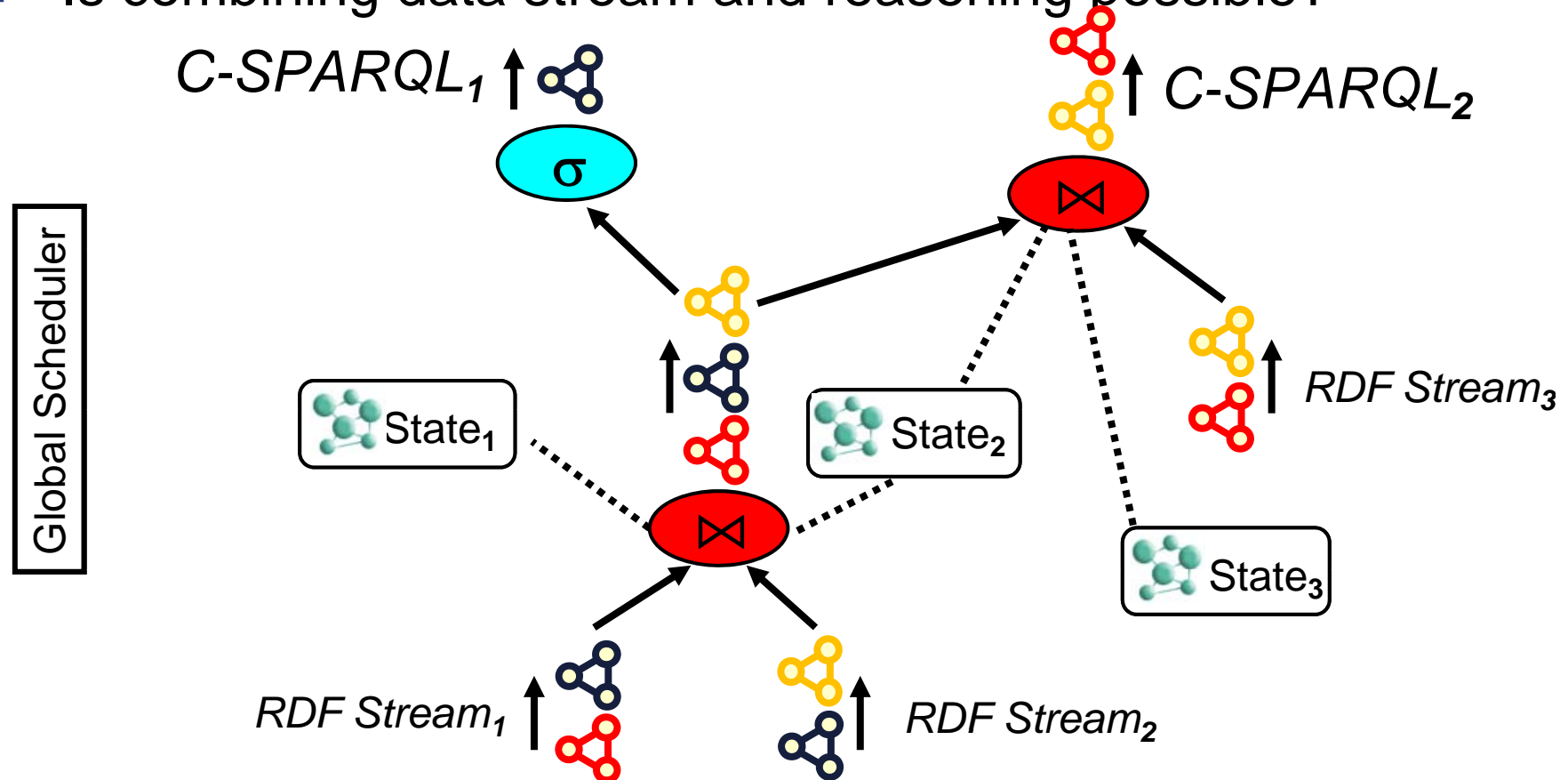
DSMS Implementations



- Research Prototypes
 - Amazon/Cougar (Cornell) – sensors
 - Aurora (Brown/MIT) – sensor monitoring, dataflow
 - Gigascope: AT&T Labs – Network Monitoring
 - Hancock (AT&T) – Telecom streams
 - Niagara (OGI/Wisconsin) – Internet DBs & XML
 - OpenCQ (Georgia) – triggers, view maintenance
 - Stream (Stanford) – general-purpose DSMS
 - Stream Mill (UCLA) - power & extensibility
 - Tapestry (Xerox) – publish/subscribe filtering
 - Telegraph (Berkeley) – adaptive engine for sensors
 - Tribeca (Bellcore) – network monitoring
- High-tech startups
 - Streambase, Coral8, Apama, Truviso
- Major DBMS vendors
 - are all adding stream extensions as well

Is Stream Reasoning possible?

- Is combining data stream and reasoning possible?

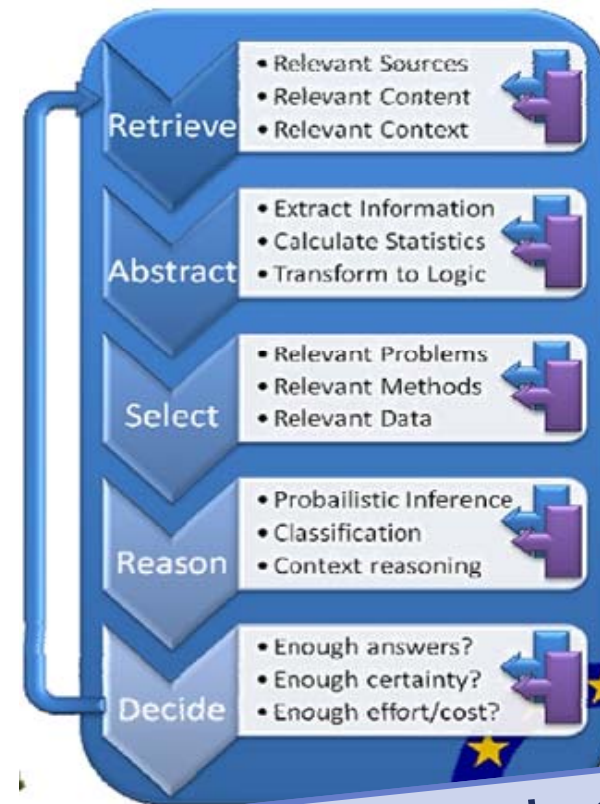
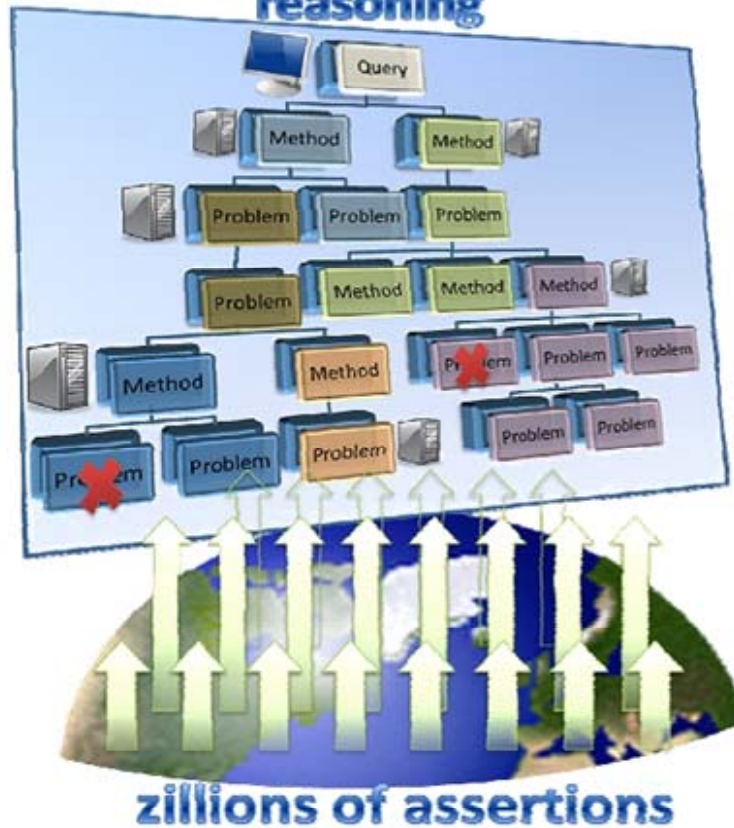


- Can the innovation so far conned within the DB community be leveraged in realizing a new generation of reasoners able to cope with continuous reasoning tasks?

A Conceptual Architecture for Stream Reasoning 1/2

- We are developing the Stream Reasoning vision with the LarkC European Research Project

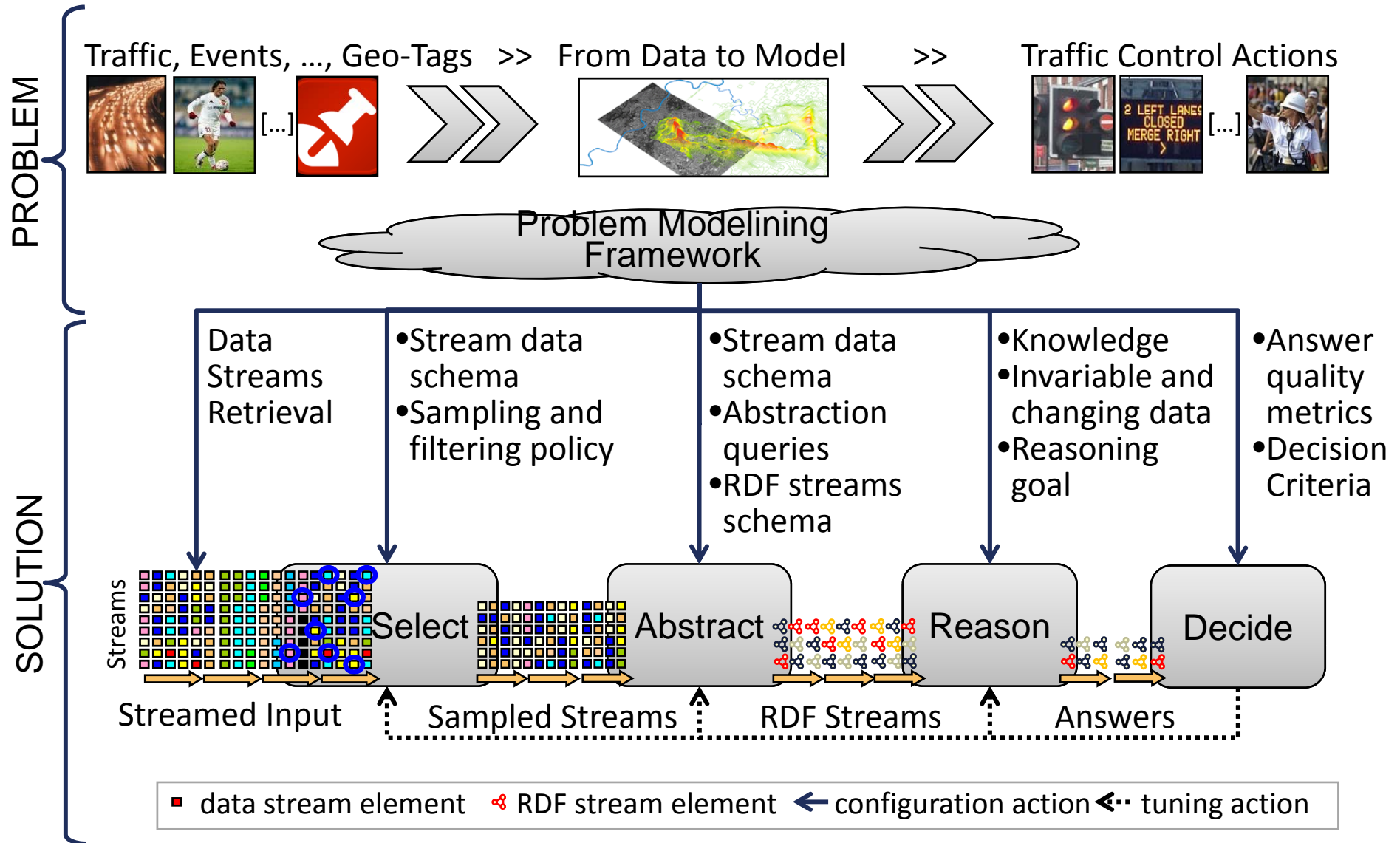
massive distributed incomplete reasoning



Visit <http://www.larkc.eu> !

[Source: Fensel, D., van Harmelen, F.: Unifying reasoning and search to web scale. IEEE Internet Computing 11(2) (2007)]

A Conceptual Architecture for Stream Reasoning 2/2



Conceptual Architecture: concepts



- On top, the problem space is grounded in the Urban Computing scenario, below, input is provided to the "LarkKC steps"
- **Retrieval** is intentionally left aside (nothing specific to streams)
- **Selection** mostly applies load-shedding techniques
 - Explicit, implicit, or learned/inferred sampling and filtering policies
- **Abstraction** shifts fine grain raw data streams into "aggregate events"
 - By means of data compression techniques (histograms, wavelets)
 - By means of aggregation operators, Bloom filters, ...
 - Abstraction is responsible for "lifting" raw data into RDF
 - typically in the form of **streams of RDF data** (details next)
- **Reasoning** depends on the notion of RDF stream
- **Decisions** are propagated back to the pipeline so as to on-line tune the behavior
 - Adaptive sampling, grouping/aggregation, graceful degradation, ... in order to let the system continue to catch up with the real-time requirements
 - Quality metrics for answers and decision criteria are hopefully provided by application designers

The two key ingredients



■ RDF streams

- **new data formats** set at the confluence of conventional data streams and of conventional atoms usually injected into reasoners

■ Continuous SPARQL (**C-SPARQL**)

- The distinguishing feature of C-SPARQL is the support for **continuous queries**, i.e. SPARQL-like queries registered over RDF data streams in the context of a C-SPARQL execution environment and then continuously executed

What is an RDF stream?



- Only a complete **RDF molecules** is stream element
- Molecules are the smallest components obtained from lossless RDF graph decomposition [Ding&Al2005]

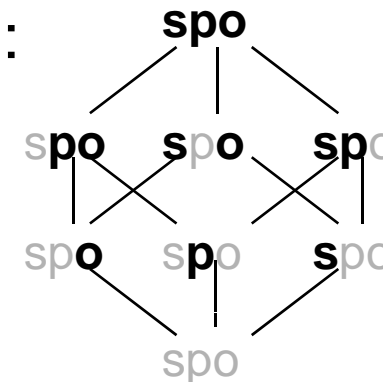
<code>uc:sensor42 uc:measure _:x .</code>	27.9.2008-11.30.00
<code>_:x uc:numberOfCars "120" .</code>	
<code>_:x uc:numberOfTrucks "70" .</code>	
<code>_:x uc:numberOfOtherVehicles "37" .</code>	

- Every new **RDF statement** (triple) is a stream element

<code>uc:sensor42 uc:measure _:x .</code>	27.9.2008-11.29.58
<code>_:x uc:numberOfCars "120" .</code>	27.9.2008-11.29.59
<code>_:x uc:numberOfTrucks "70" .</code>	27.9.2008-11.30.00
<code>_:x uc:numberOfOtherVehicles "37" .</code>	27.9.2008-11.30.01

- Classification of RDF statement stream:

- From fully bound to fully free
 - Free
 - bound subject, predicate, object
 - free subject, predicate, object
 - bound



Two approaches to stream reasoning



■ Evolutionary approach

- Reuse of existing technology
- Based upon adapters
- Streams of **RDF molecules**
- Fast prototyping possible

■ Revolutionary approach

- Development of radically new concepts and design of revolutionary technologies
 - A specific query language and time-aware reasoners
- **Streams of RDF statements**
- Research framework

Two approaches to stream reasoning



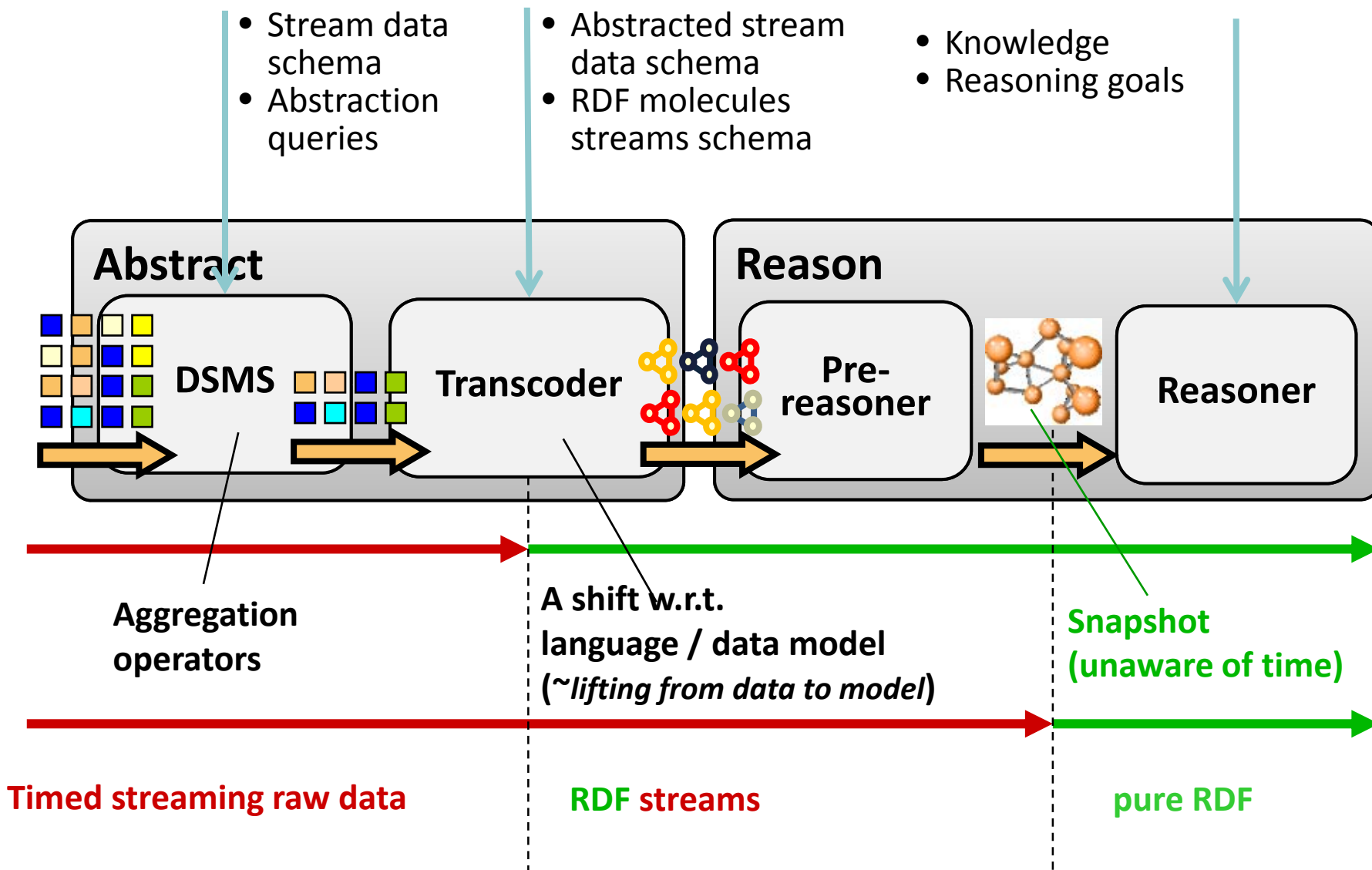
■ Evolutionary approach

- Reuse of existing technology
- Based upon adapters
- Streams of **RDF molecules**
- Fast prototyping possible

■ Revolutionary approach

- Development of radically new concepts and design of revolutionary technologies
 - A specific query language and time-aware reasoners
- Streams of **RDF statements**
- Research framework

Evolutionary approach



The evolutionary approach requires...



- **Use of existing technologies for the DSMS and the reasoner**
 - Massive use of CQL for selection purposes
 - But it works on raw data
 - Careful, semantic stream registration
- **Transcoding and pre-reasoning** for extracting subsequent snapshots of information
 - **Incremental maintenance of snapshots**
 - These are embedded in the abstraction & reasoning steps of LarKC

Two approaches to stream reasoning



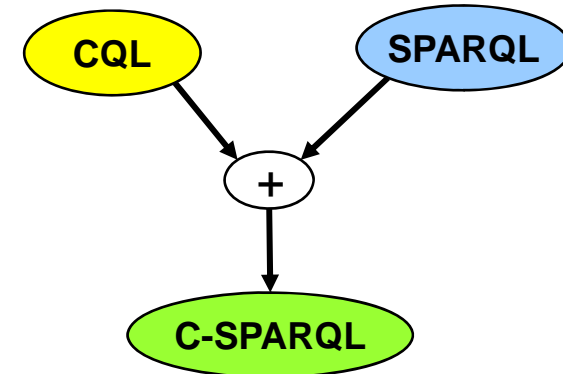
- **Evolutionary approach**
 - Reuse of existing technology
 - Based upon adapters
 - Streams of **RDF molecules**
 - Fast prototyping possible

- **Revolutionary approach**
 - Development of radically new concepts and design of revolutionary technologies
 - A specific query language and time-aware reasoners
 - **Streams of RDF statements**
 - Research framework

Revolutionary approach: C-SPARQL



- Reasoning on streams with a paradigm which includes streams overall in the reasoning process
- **Current focus** on inventing **C-SPARQL**
 - requirements
 - applicability
 - expressive power/efficiency trade-offs
 - comparison with existing work
- **A query** in C-SPARQL



```
REGISTER STREAM new_pirates COMPUTED EVERY 1 SEC AS  
CONSTRUCT { ?vehicle a uc:Pirate .  
              ?vehicle uc:RecentViolations ?countViolations }  
FROM STREAM <http://uc.larkc.eu/highspeed.trdf>  
      [RANGE 1 HOURS STEP 1 MIN]  
WHERE { ?vehicle uc:hasHighSpeedOn ?street .  
          ?v a uc:Pirate .  
          FILTER (?v != ?vehicle) }  
AGGREGATES {(?countViolations, COUNT, ?vehicle) .  
              FILTER (?count > 5)}
```

Research agenda (medium term)



- **Syntax** (full specification of C-SPARQL)
 - Incrementally, from existing specifications
 - Including windows, grouping, aggregates, timestamping
 - Distinguishing streams of triples vs molecules
- **Semantics** (Formal semantics of C-SPARQL)
 - Query registration, handling overloads
 - Order of evaluation, pattern matching over time, ...
- **Technology** (Efficiency of evaluation)
 - Defining a suitable algebra
 - Efficient materialization of inferred data from streams
 - Time window-dependant KB maintenance
 - Smart indexing techniques that benefit from the FIFO semantics (timestamp-based obsolescence of inference chains)

...exploiting parallelism (long term)



- **Streams are parallel in nature**
 - Keep them separate as long as possible, and push sampling and filtering (selection/abstraction) near to the source
- **1st level of parallelism (inter-stream)**
 - Each stream can be handled by dedicated processors
 - Each operator can be handled by a different CPU
- **2nd level of parallelism (intra-stream)**
 - Windowing may occur in parallel
 - Different CPUs address different (possibly overlapping) windows over the same stream
 - overlapping windows can be divided into disjoint panes, to efficiently compute sub-aggregates over each pane.

Thank you for paying attention



Any Questions?



Politecnico di Milano
Dipartimento di Elettronica e Informazione

&



A First Step Towards Stream Reasoning

Lecturer:

Emanuele Della Valle

emanuele.dellavalle@polimi.it

<http://emanueledellavalle.org>

Authors:

Emanuele Della Valle, Stefano Ceri, Davide F. Barbieri,
Daniele Braga and Alessandro Campi

