Behavior Protocols

SOFA and beyond

Frantisek Plasil

DISTRIBUTED SYSTEMS RESEARCH GROUP

http://nenya.ms.mff.cuni.cz

CHARLES UNIVERSITY PRAGUE

Faculty of Mathematics and Physics

School of Computer Science

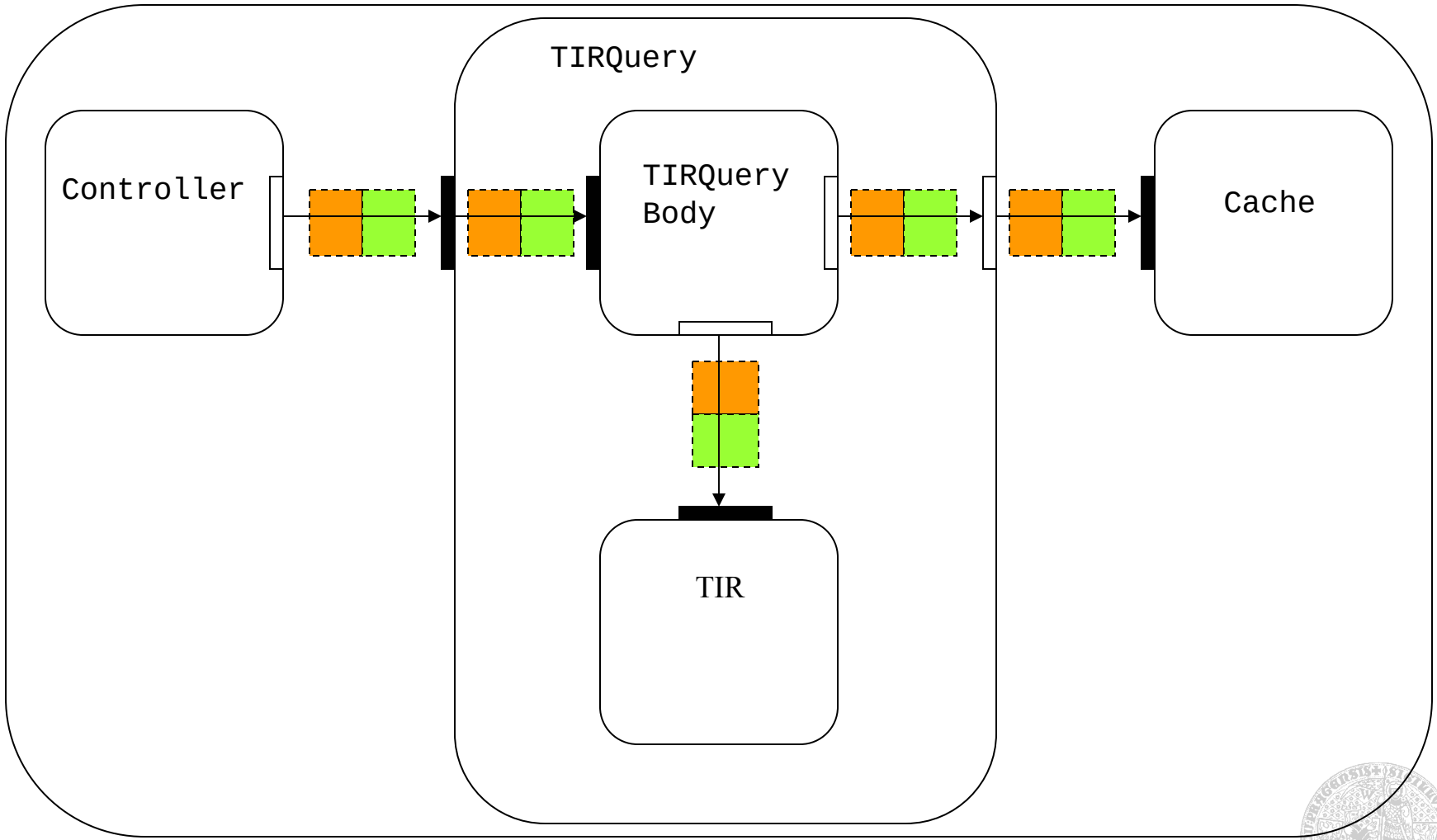
Outline

- Motivation
  - SOFA component model overview
- Inspiration
  - Some milestones
- Power of behavior protocols
  - Compliance +composition errors + …
  - Rum time checking
- Behavior protocols' property checking
  - A specific method(s) of model checking

Motivation

- SOFA component model
  - Software Appliances
  - OMG compliant deployment framework
    - Heterogeneous: Connectors to other component models
      - Fractal,
      - EJB
  - Open source, available at: **www.objectweb.org**

TIRBrowser SOFA example



Component Model SOFA

- Hierarchical components
- Provides/requires interfaces with ties
  - Binding
  - Subsume
  - Delegation
- Connectors
  - ProcCall
  - EventPassing
  - DataStream
  - *User defined*
- Protocols

CDL spec Behavior protocols

```
interface TIRAccessInterface {
  void init();
  string query(in string
   name);
  void finish();
  protocol:
    init; (query)*; finish
};

frame TIRQueryBody {
  provides:
    TIRQueryInterface query;
  requires:
    TIRAccessInterface tir;
    CacheInterface cache;
  protocol:
   !tir.init;
   ?query.query{ !cache.get;
        (!tir.query+NULL) }* ;
   !tir.finish
};
```

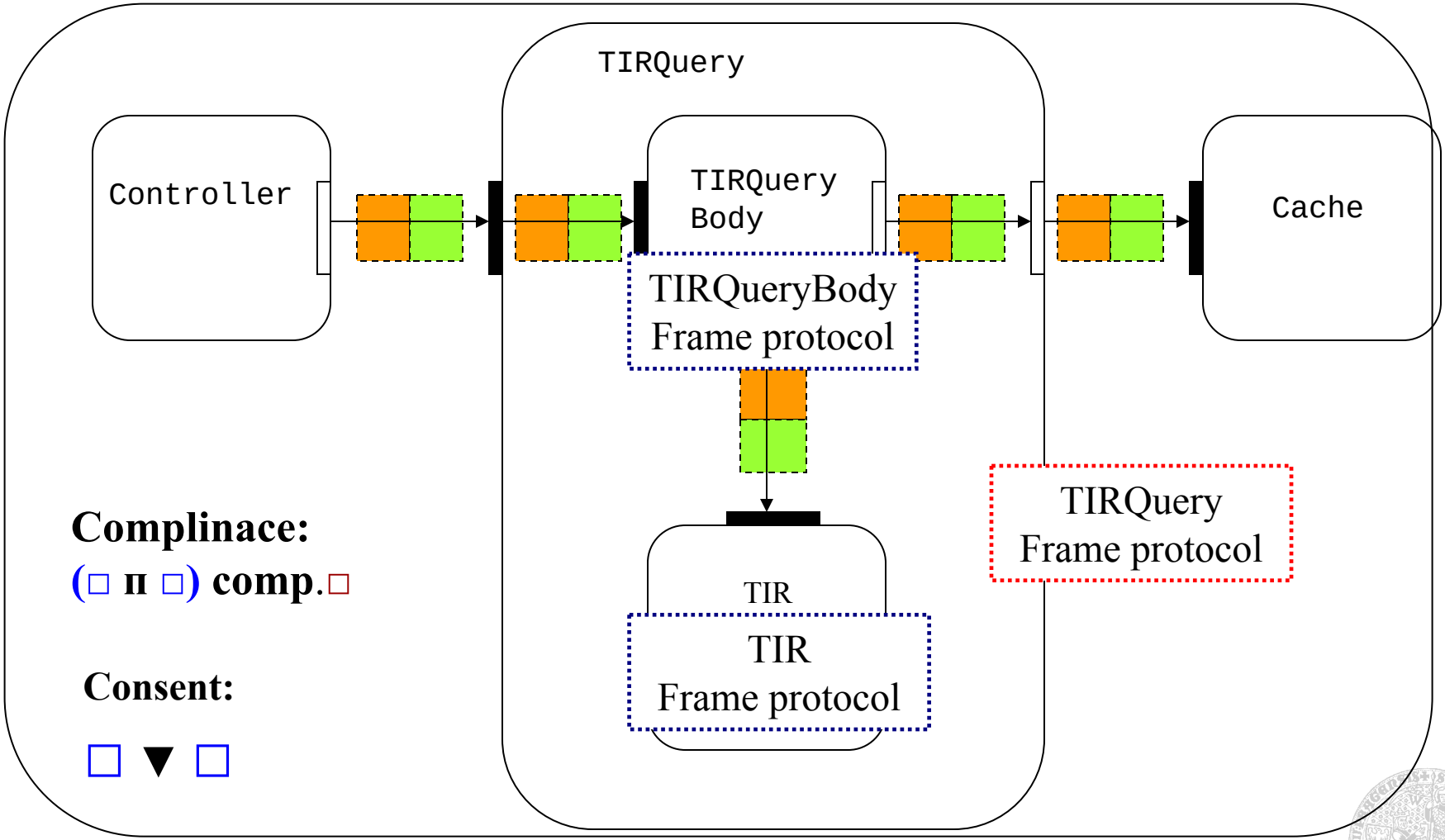
```
frame TIRQuery {
  ...
   protocol ...
}

architecture CUNI TIRQuery
     implements TIRQuery {

  inst TIRQueryBody body;
  inst TIR tir;

  bind body:tir to tir:access;
  delegate query to
   body:query;
  subsume body:cache to cache;
   -- protocol: auto generated
}
```

Protocol checks Compliance & Consent



Applying protocols: Checks options

- Design
  - Vertical: Adjacent levels of abstraction
    - ***Do the cooperating children do what the parent expects***?
    - **Compliance**
  - Horizontal: a specific level of abstraction
    - ***Do the children cooperate with no conflict?***
    - **Consent**
- Runtime
  - Checking real behavior against protocols
    - **Runtime checker**

As an aside

- CDL spec *By hand*
- Compiling CDL
  - Check Protocol Compliance & Consent
  - Generating Code fragments
    - Types
    - Component builders
    - Assembly descriptors
- Primitive components
- Assembling
  - Filling out Assembly descr.
- Deploying application to virtual nodes
  - Filling out Deployment map
  - Connector generation
- Launching
  - Opportunity: Checking real behavior against protocols

Behavior protocols — Pros

- LTS defined via expressions
  - `(open ; (read + write)* ; close) | status*`
  - ***Easy to read & comprehend***

- Properties of the state space
  - Compliance
  - Composition errors
  - Incomplete bindings
  - Update atomicity
  - ***But not to implement***
    - ***Model checking***

Inspiration & Aim

- Background
  - Path expressions
  - CSP + CCS
  - LOTUS
  - …
  - Model checking
- Motivation & Aims
  - Readability and easy comprehension of LTS spec
  - Decidability and chance to handle state explosion in “reasonable” practical limits

Technical comments

- Events on connections
- Reminds regular expression
- Addition: and and or parallel operator

- Abbreviations
- Semantics: traces (could infinite)

Evolution

Milestones

- Tools US'99
  - Naive compliance: (~ simple subtyping = replacement)

    New: "More on provides, less on requires"
- IEEE Trans 2002
  - *Plasil F, Visnovsky S. Behavior protocols for Software Components. IEEE Trans. on SW Engineering, 28(9), 2002*.
  - Pragmatic compliance

    New: On a specific input sequence, at lest one reaction of those of the old + plus able to react on all the inputs accepted by the old.

- *Adamek, Plasil: Journal of SW Maintenance, 2003, +…*
  - Composition Errors and update atomicity
  - Incomplete bindings
  - Faulty architecture is a relative concept
  - Consensual compliance

Protocol Compliance

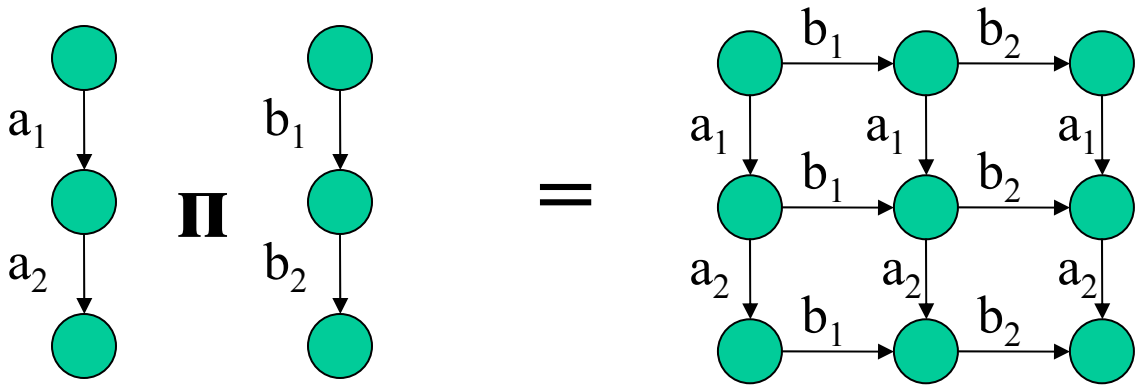
- Protocol A is ***compliant*** to protocol B, if A can be used instead of B, i.e:
  1) $L(B)/S_{prov} \subseteq L(A)/S_{prov}$
  2) $L(B)/S_{prov\ |S_{prov}|}\ L(A) \subseteq L(B)$

- Compliance check
  - trace sets inclusion, i.e.

    $A \cap compl B = O$

Problem: State Space Explosion

- Blow-up of the state space
  - Parallel compositions
    - Exponential 3^2



Solutions – our choice

- Structural simplifications of the state space
  - Several levels of ***abstraction***
- Efficient state space representation
  - ***OBBD***s and their variants
    - LTS mapping to BBD (*To be found*)
- ***On-the-fly*** state space generation
  - in a lazy way
  - not fully generated
    - with optimizations

Case studies

- Air traffic control
- Capturing Use Cases
  - Pro-cases
- Speedo project analysis
  - Reverse engineering

Conclusion and Future Work

Future intentions:

- Checker: Using ***OBDD***s
  - Checking protocols with ~30-40 (??) parallel operators

- Code analysis vers. Behavior spec.

  (bottom up) (top down)

- Run time checker

Questions…?

Answers also at

http://nenya.ms.mff.cuni.cz