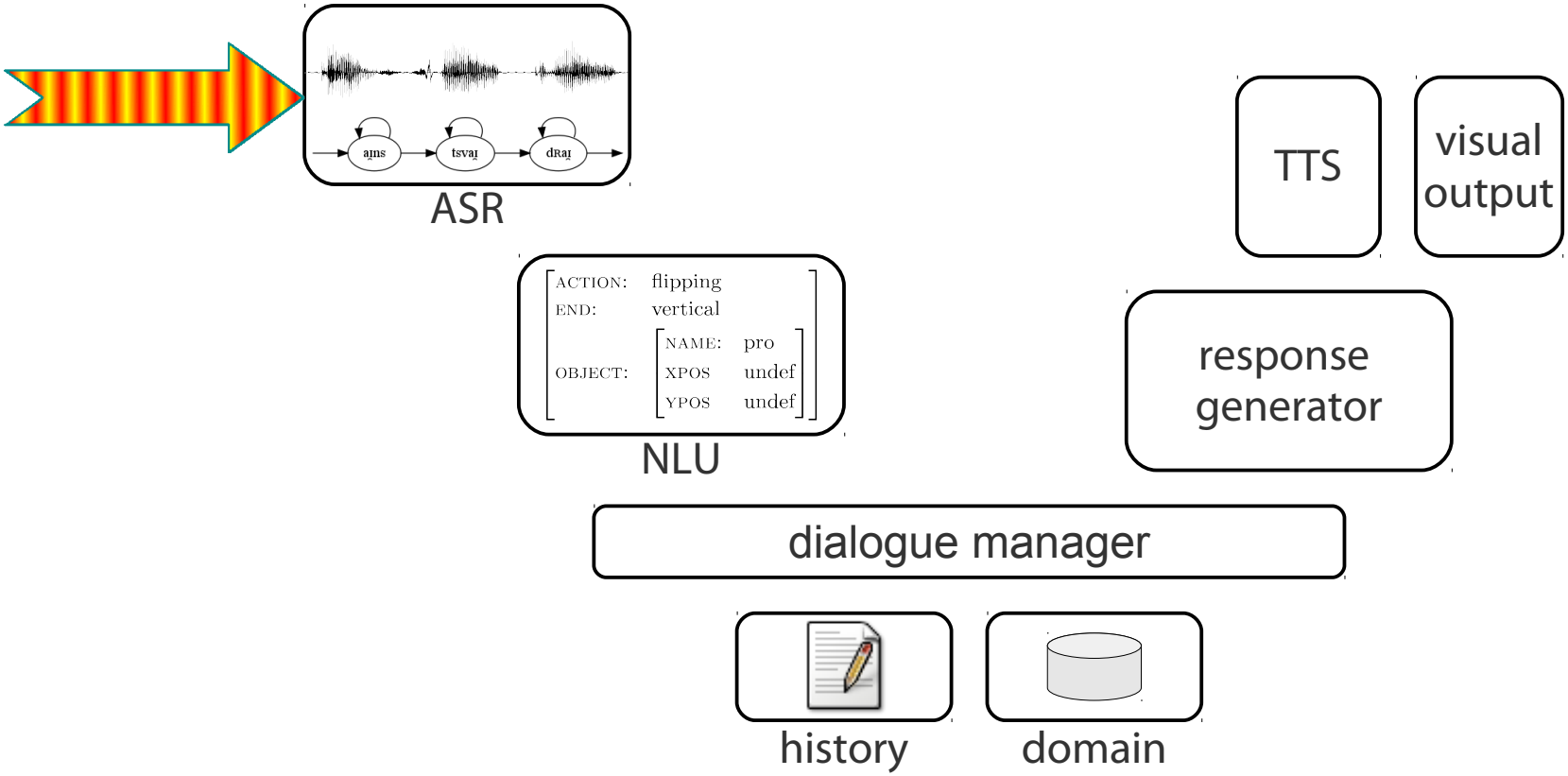
InproTK in Action: Open-Source Software for Building German-Speaking Incremental SDSs

Timo Baumann, Okko Buß, David Schlangen

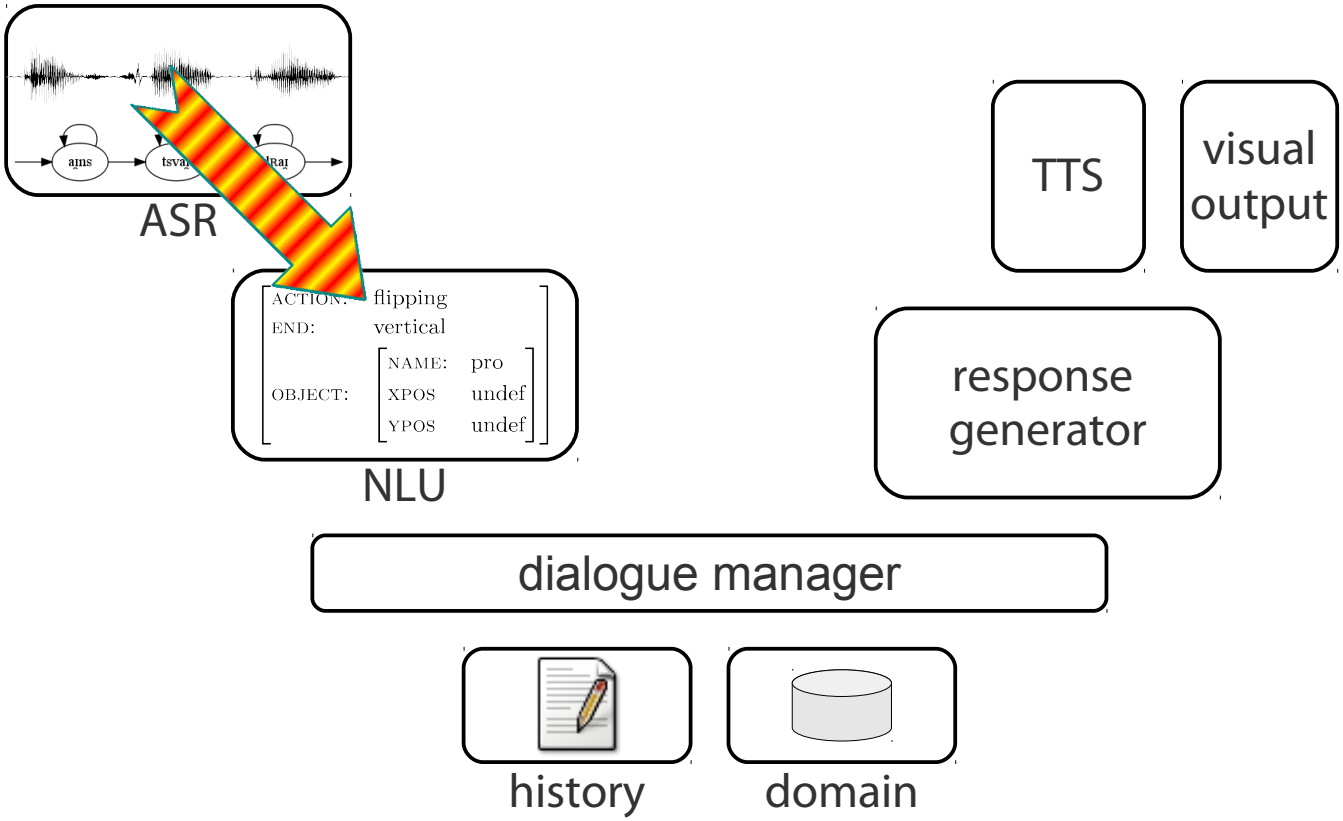
timo@ling.uni-potsdam.de

http://www.ling.uni-potsdam.de/~timo

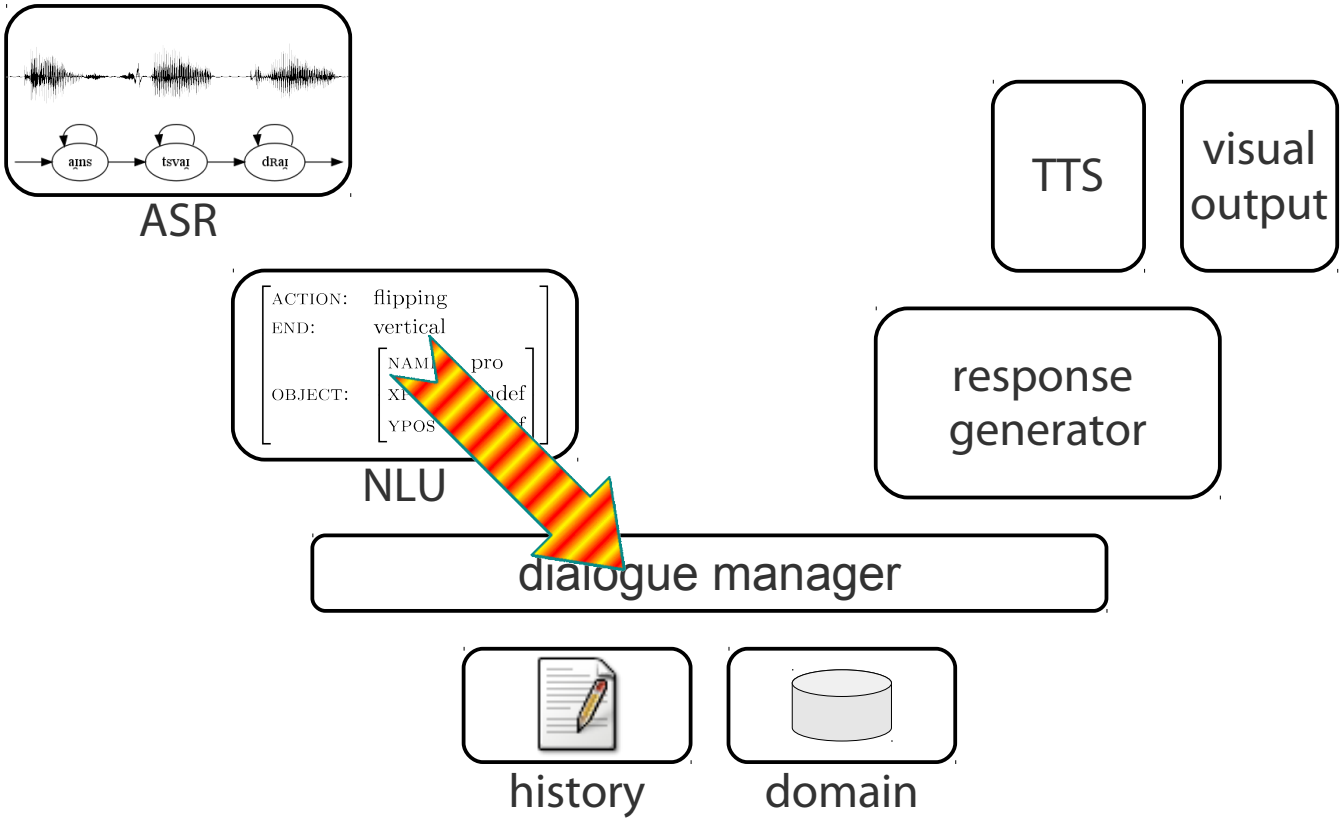
Spoken Dialogue Systems



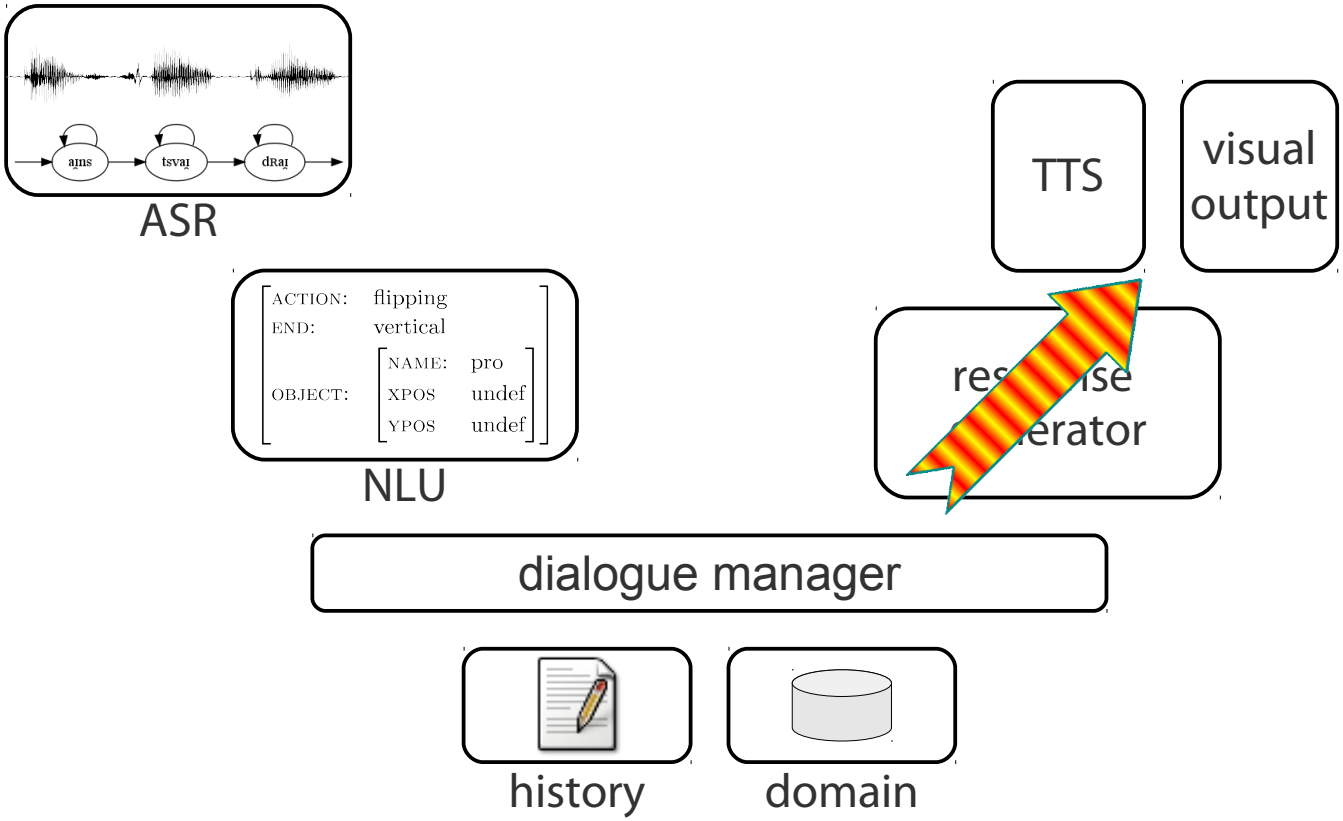
Spoken Dialogue Systems



Spoken Dialogue Systems

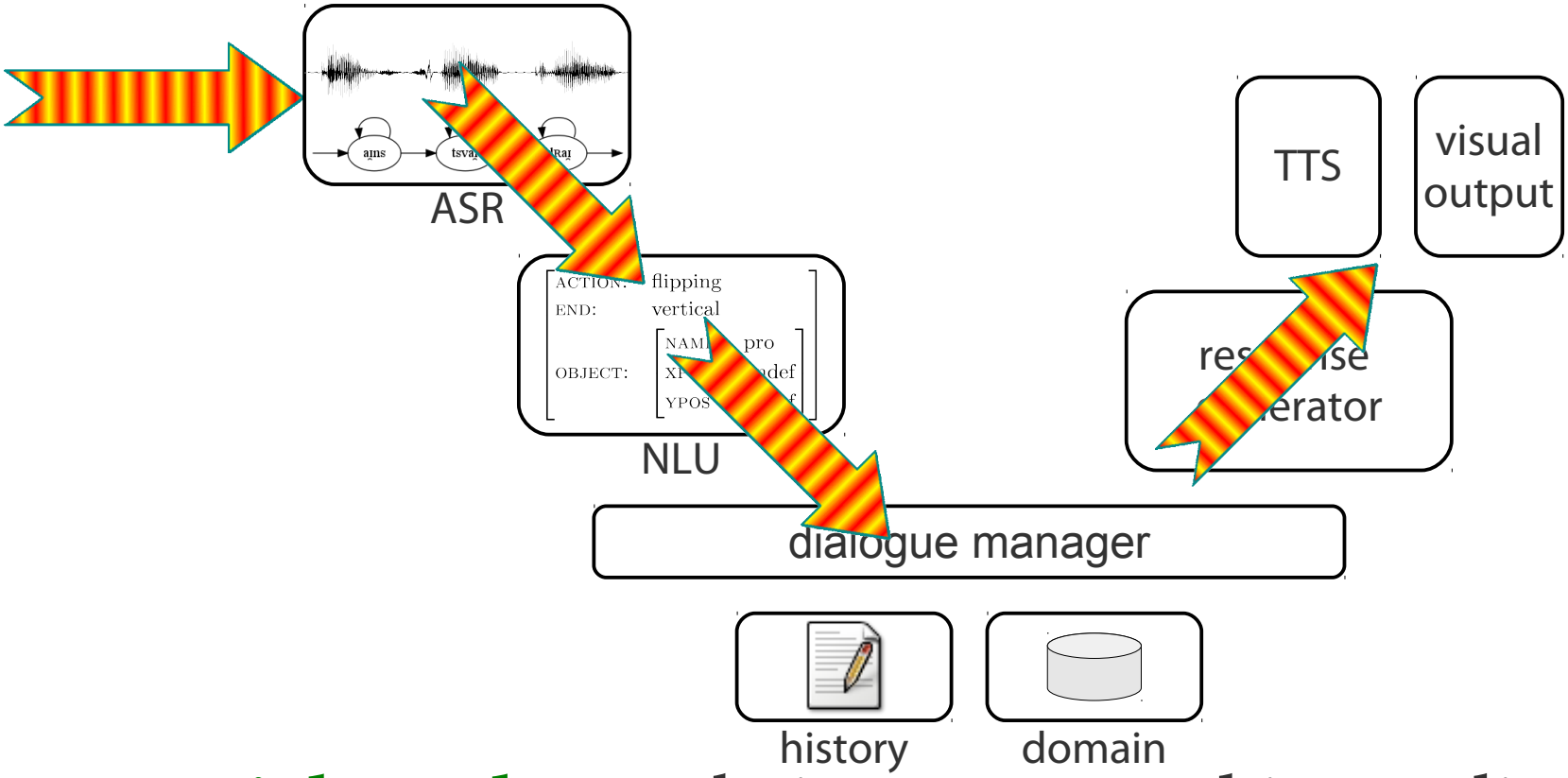


Spoken Dialogue Systems



- modules start after their predecessors have finished

Incremental
Spoken Dialogue System



- **partial results** are being processed immediately
- reaction is quicker, interaction more natural

Benefits of Incremental Spoken Dialogue Systems

1. react more quickly
   as modules process input during a speaker's turn:

U: Ich möchte am Samstag von Berlin nach Hamburg fahren.

S: Ok, um wieviel Uhr möchten Sie fahren?

(Crafted examples for an imaginary train timetable information system.)

Benefits of Incremental Spoken Dialogue Systems

1. react more quickly
   as modules process input during a speaker's turn:

U: Ich möchte am Samstag von Berlin nach Hamburg fahren.

S: Ok, um wievi[illegible] [illegible]r möchten Sie fahren?

sufficient information:
Saturday, B → HH

Benefits of Incremental Spoken Dialogue Systems

1. react more quickly
   as modules process input during a speaker's turn
2. give feedback during a speaker's turn:

```
U: Ich möchte am Samstag mit dem ICE
   Nummer, äh ... warten sie ... 798 ...
S:                ja?                ok.
```

- feedback might be visual in a multi-modal system

Benefits of Incremental Spoken Dialogue Systems

1. react more quickly
   as modules process input during a speaker's turn
2. give feedback during a speaker's turn
3. even interrupt a speaker's turn:

U: Ich möchte am Samstag mit dem ICE Nummer 798 nach, äh ...

S: Entschuldigung, ICE 798 verkehrt nicht samstags, wohin möchten Sie denn fahren?

Benefits of Incremental Spoken Dialogue Systems

1. react more quickly
   as modules process input during a speaker's turn
2. give feedback during a speaker's turn
3. even interrupt a speaker's turn

→ all these capabilities make the SDS **more similar** to a human interlocutor

Content:

- ✔ Advantages of incremental SDSs
- ➔ Requirements for incremental SDSs
- Our model of incremental processing
- Our implementation: InproTK
  - Overview of the architecture
  - Predefined Modules
- Example systems

Requirements for Incremental SDSs

- System fully embraces incrementality
  - it's very hard to adapt a pre-existing SDS to turn it into an incremental system
- 100 % incremental modules
  - just one non-incremental module breaks the pipeline
- Processing delays are minimized (buffering, etc.)
  - across the board – all processing delays add up!
  - otherwise too slow for really interesting applications

Requirements (II): Dealing with Uncertainty

- intermediate hypotheses **change with time**
  - we may get things wrong intermittently:

„Hamburg" → /hamburg/

- *Incrementally* this will look to speech recognition as follows …

Requirements (II): Dealing with Uncertainty

- intermediate hypotheses **change with time**
  - we may get things wrong intermittently:

„Hamburg" → /hamburg/

- ASR:

this sounds like „Hamm"

- NLU:

they must be talking about [city:Hamm(Westfalen)]

Requirements (II): Dealing with Uncertainty

- intermediate hypotheses **change with time**
  - we may get things wrong intermittently:

„Hamburg" → /hamburg/

- ASR:

this sounds like „Hamburg"

- NLU:

they must be talking about [city:Hamburg]

Requirements (II): Dealing with Uncertainty

- intermediate hypotheses **change with time**
  - we may get things wrong intermittently:

„Hamburg" → /hamburg/

- Couldn't the ASR just lag behind a little bit?

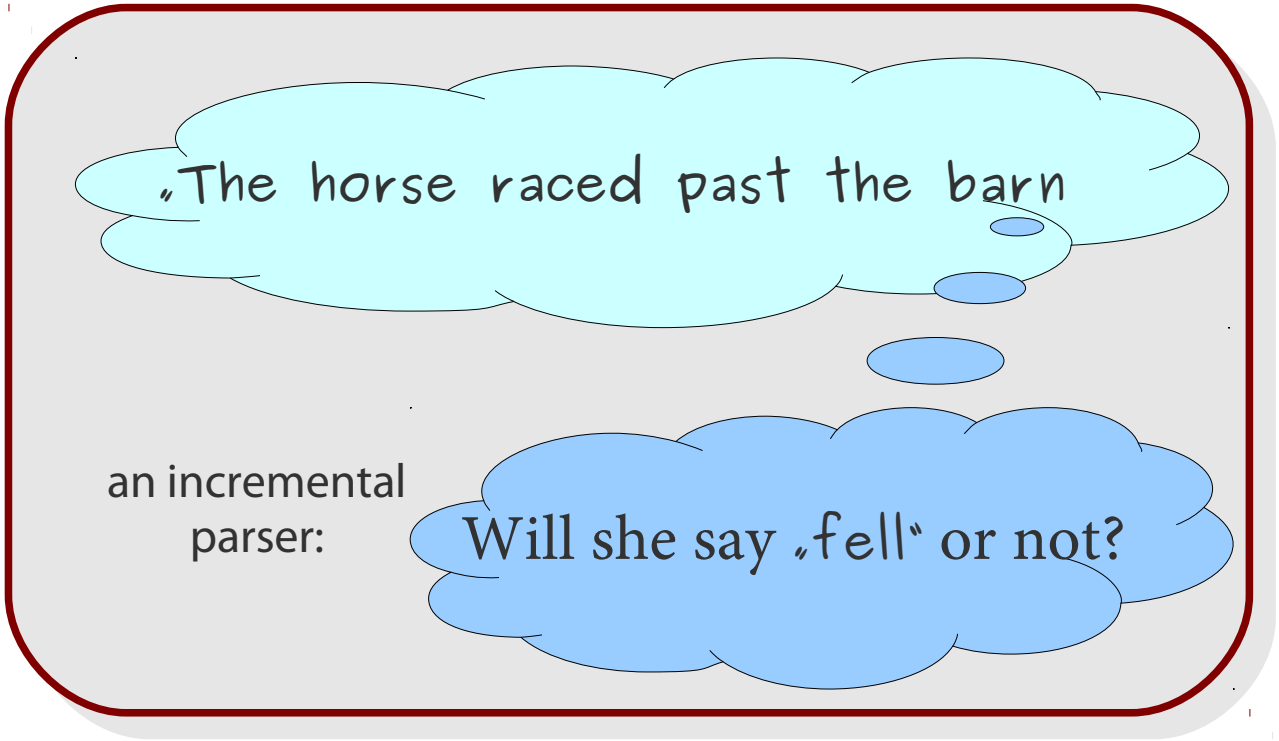
Requirements (II): Dealing with Uncertainty

- Couldn't the ASR just lag behind a little bit?
- Yes, but:
  - long-distance dependencies
  - there will always be local ambiguities
  - all delays will add up

➔ hence, previous hypotheses must be **changeable**

Requirements (II): Dealing with Uncertainty

- Couldn't the ASR just lag behind a little bit?
- Yes, but:
- e.g. garden-path sentences, …



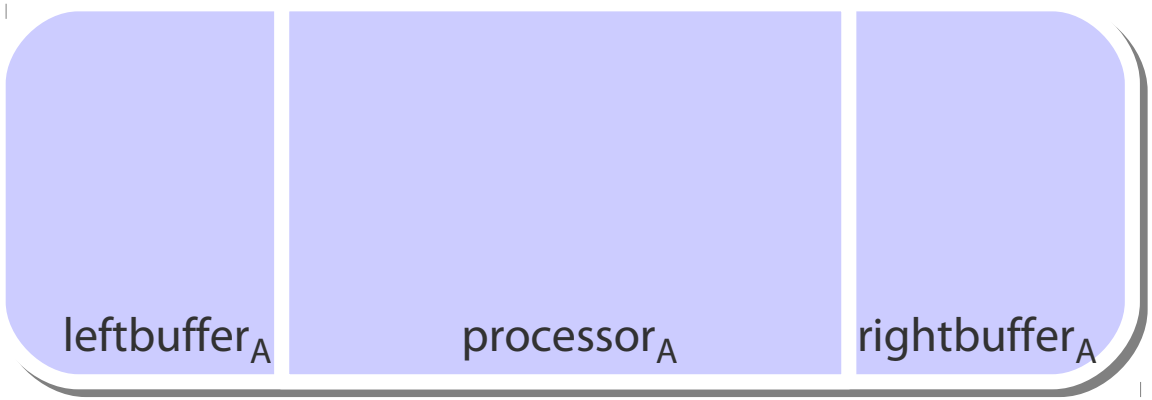
→ hence, previous hypotheses must be **changeable**

Content:

- ✔ Advantages of incremental SDSs
- ✔ Requirements for incremental SDSs
- ➔ Our model of incremental processing
- Our implementation: InproTK
  - Overview of the architecture
  - Predefined Modules
- Example systems

Our Model of Incremental Processing

- A system consists of several connected modules
- ***Incremental Modules*** are composed of
  - a ***left buffer***, a ***processor***, and a ***right buffer***

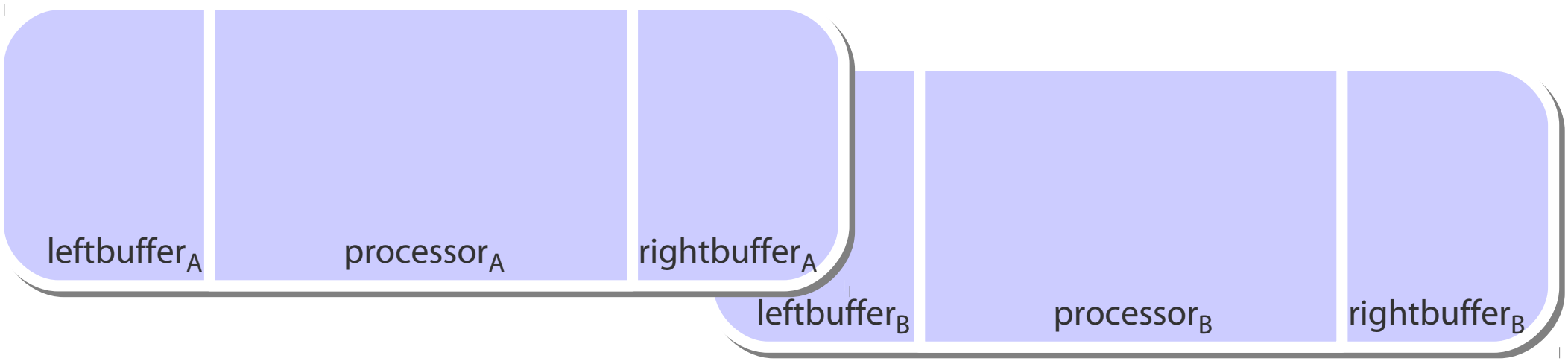


- a processor takes input from the left buffer and provides output in its right buffer

(Schlangen and Skantze, 2009)

Inter-Module Communication

- A module's right buffer may be superimposed to other modules' left buffer to share the same content

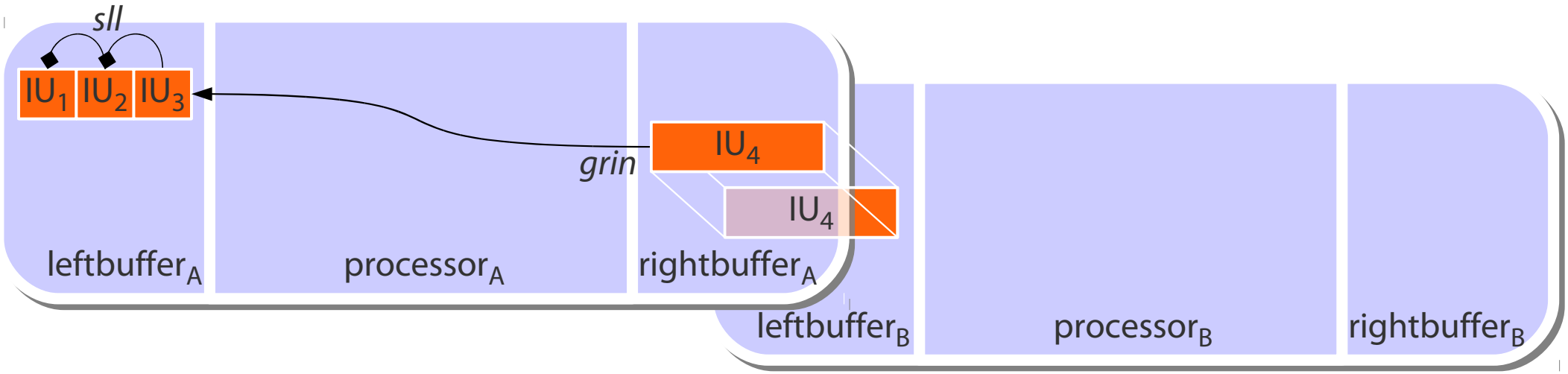


- modules communicate by probing content and adding **content** in their buffers

(Schlangen and Skantze, 2009)

Incremental Units

- Content is shared in the form of **Incremental Units** (IUs), which are smallest 'chunks' of information



- Links between IUs:
  - **grounded-in** links (*grin*) to denote ancestry
  - **same-level** links (*sll*) for information of the same type

(Schlangen and Skantze, 2009)

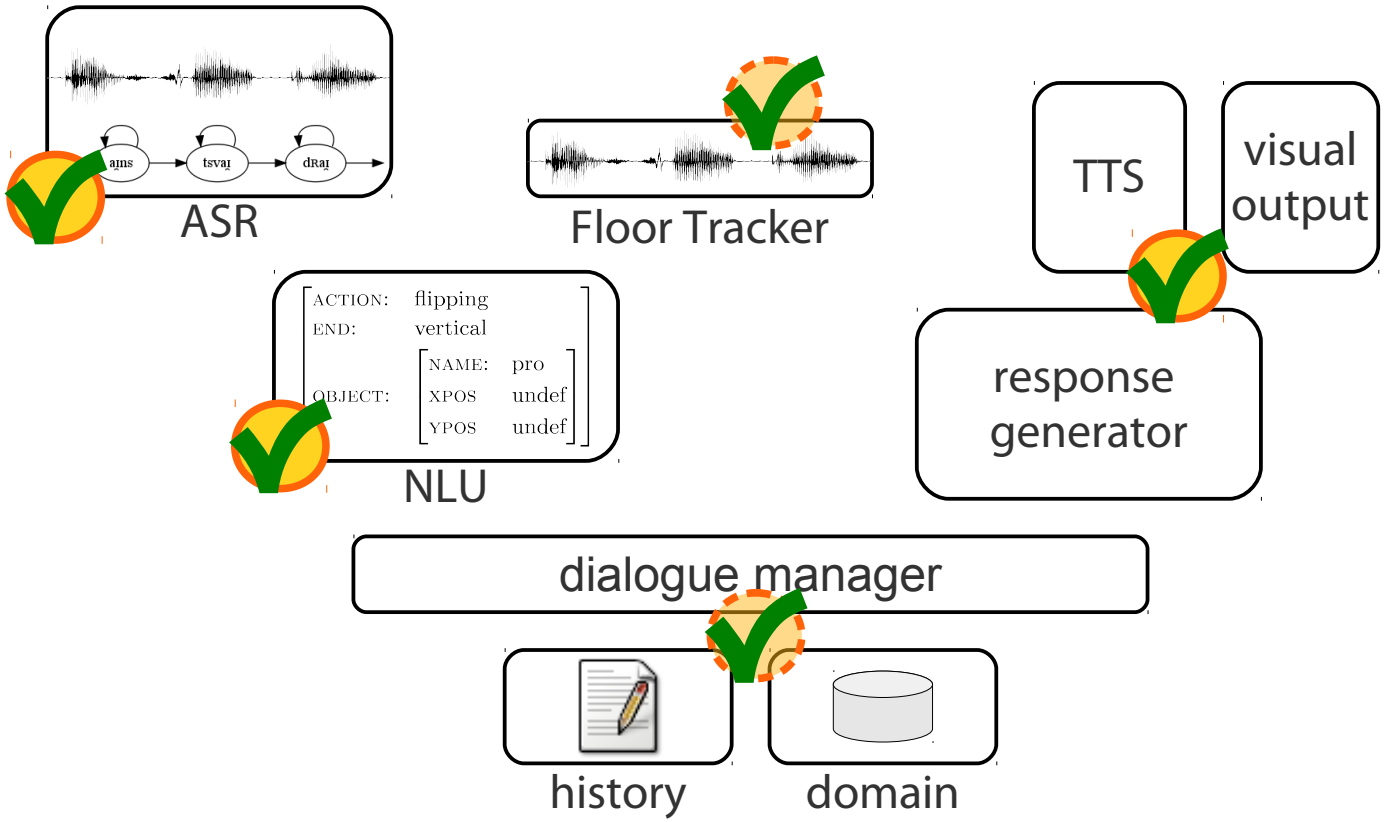
IU Network

- all IUs are connected through (*sll* and *grin*) links
  - this network contains all the information believed by the system at a certain point in time
  - the network is highly dynamic, with *changes* to the network reflecting the system's internal state *over time*
- Modules react to three basic changes:
  - new IUs are **added**
  - erroneously hypothesized IUs are **revoked**
  - IUs are **committed**, i.e. won't be changed anymore

InproTK: Overview

- Our toolkit InproTK is an implementation of our model of incremental processing
  - modular architecture
  - event-based communication between modules
- written in JAVA, integrated with Sphinx-ASR
  - rich speech recognition, prosodic processing
- extensible, open-source, somewhat documented
- www.ling.uni-potsdam.de/~timo/code/inprotk/

InproTK: Available Modules



monitoring, debugging, and analysis components

InproTK: Incremental ASR

- integrates with Sphinx-4
  - supports JSGF-grammars, SLMs, forced-alignment …
  - input from microphone, file, RTP
- current hypothesis is updated after every frame of audio consumed by the recognizer
  - hypothesis smoothing to reduce „jitter“ at the cost of some timeliness
- (show video)

(Baumann et al., 2009)

InproTK: Floor-Tracking

- turn-taking is (almost) trivial in conventional SDS
  - the user's turn is over when she stops for 500 ms
- in the incremental case, we want to be quick when we can, but not interrupt when we shouldn't

➔ a specific component that handles this complexity

  - the floor tracker emits signals like „end of turn (rising/falling/…)“, „user is holding“, „BC opportunity“, etc.
  - the dialogue manager consumes these signals

InproTK: Incremental NLU

- words are assigned *attribute-value pairs* (AVPs)
- complex semantics are represented as *attribute-value matrices* (AVMs)
- first step: composing AVPs to underspecified AVMs
- second step: resolving AVMs against (fully specified) entities in the domain

InproTK: Dialogue Management

- *information-state update* (ISU) mechanism
- based on *questions under discussion* (QUD)
- IS combines semantic slots, action planning and information grounding
- this is very much work in progress

- also, there is a simple Echo Dialogue Manager

(Buss and Schlangen, 2010)

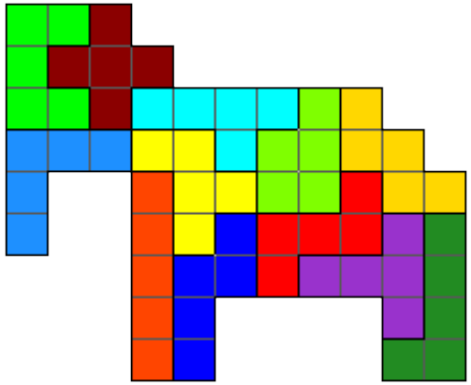
Example Application:

- show video 1

Conclusion

- I hope to have convinced you that …
  - incremental processing is vital
    for more natural dialogue systems
  - implementing such systems
    is a worthwhile endeavour
  - you should go ahead and build one yourself
    … preferably using our toolkit!

Thank you!



Acknowledgements:

Okko Buß and David Schlangen, my collaborators.
DFG for funding (Emmy Noether programme)