SIGDial 2012

Combining Incremental Language Generation and Incremental Speech Synthesis for Adaptive Information Presentation

Hendrik Buschmeier*, **Timo Baumann****,
Benjamin Dorsch*, Stefan Kopp*, David Schlangen*

*U Bielefeld, **U Hamburg, Germany

Combining **Incremental Language Generation** and **Incremental Speech Synthesis** for Adaptive Information Presentation

→ Incremental **Speech Output**

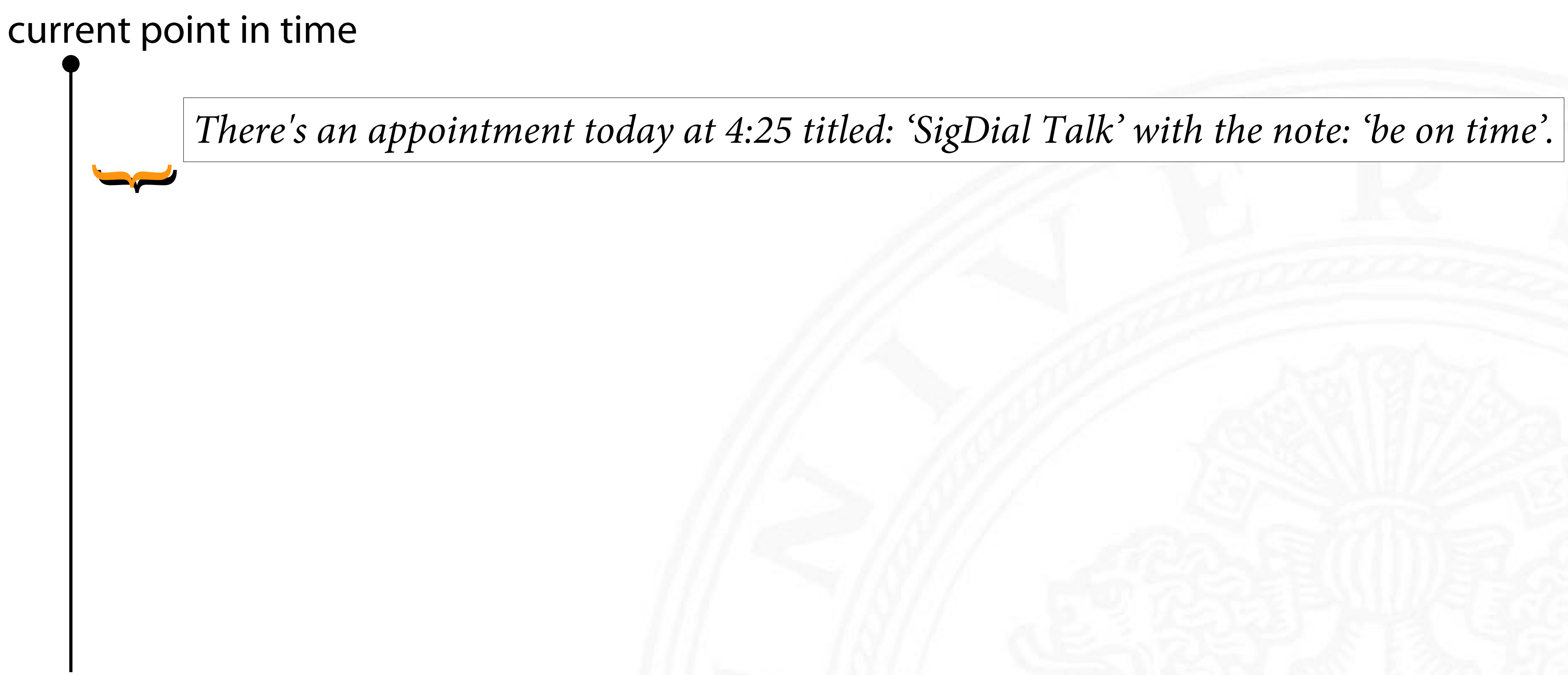
Speech Output in Typical Dialogue Systems

current point in time

There's an appointment today at 4:25 titled: 'SigDial Talk' with the note: 'be on time'.

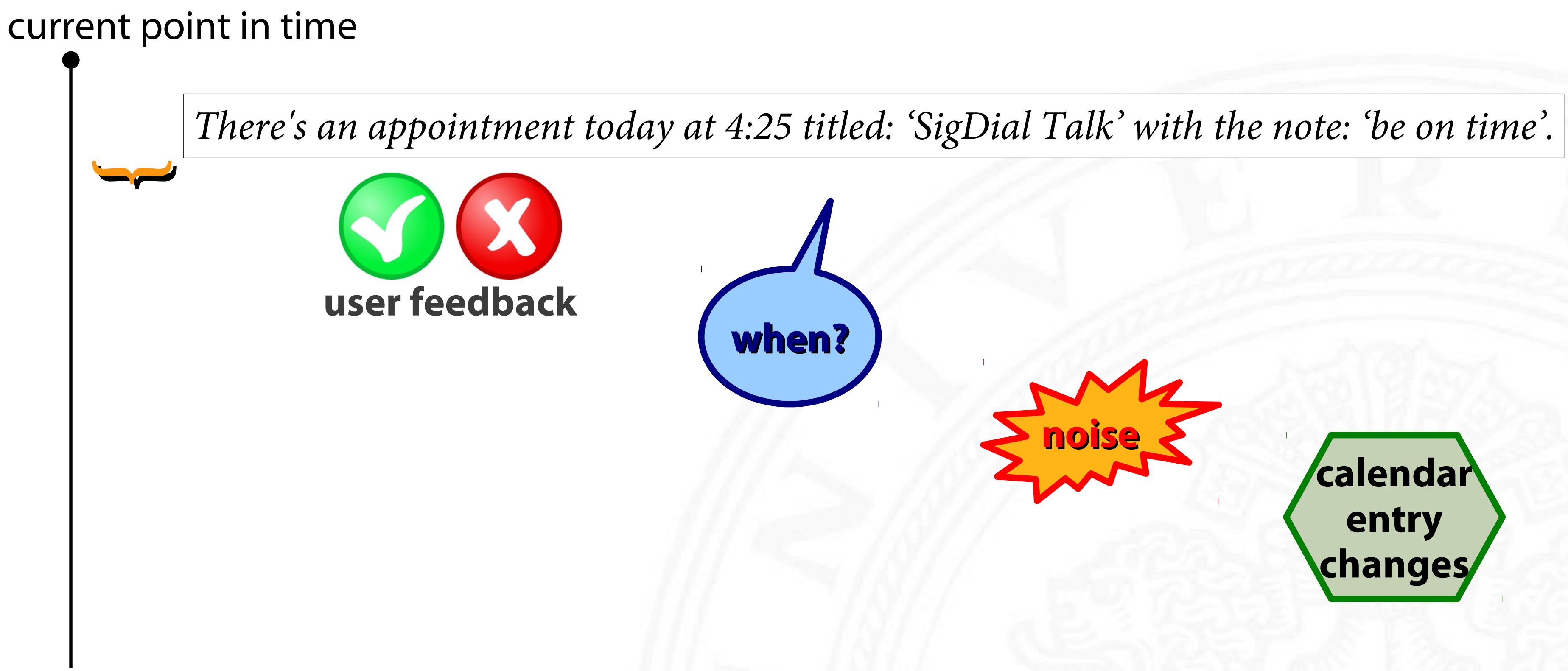
- full utterances are generated, synthesized and delivered as a whole

Speech Output in Typical Dialogue Systems



- potentially slow, as all processing is utterance-initial
  - canned speech in deployed systems

Speech Output in Typical Dialogue Systems



- inflexible: unable to change the ongoing utterance
  - no way to react to the listener or the environment

Potentially Better:
Incremental Speech Output

current point in time

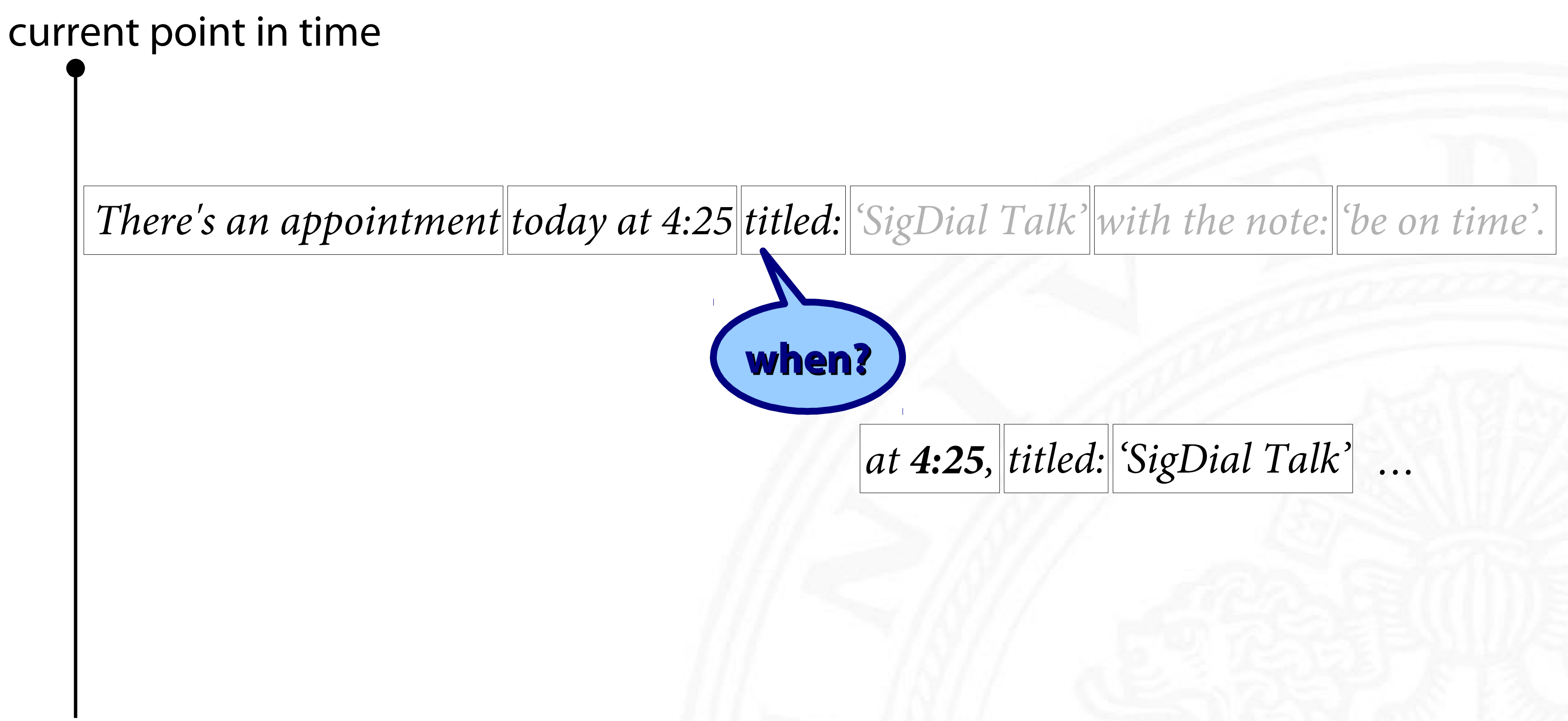
There's an appointment | *today at 4:25* | *titled:* | *'SigDial Talk'* | *with the note:* | *'be on time'.*

- generate, synthesize and deliver the utterance in smaller *chunks*

Potentially Better:
Incremental Speech Output

- less utterance-initial processing → faster onset

Potentially Better: Incremental Speech Output



- incremental output may take changes into account
- react and adapt to user feedback / requests / **noise**

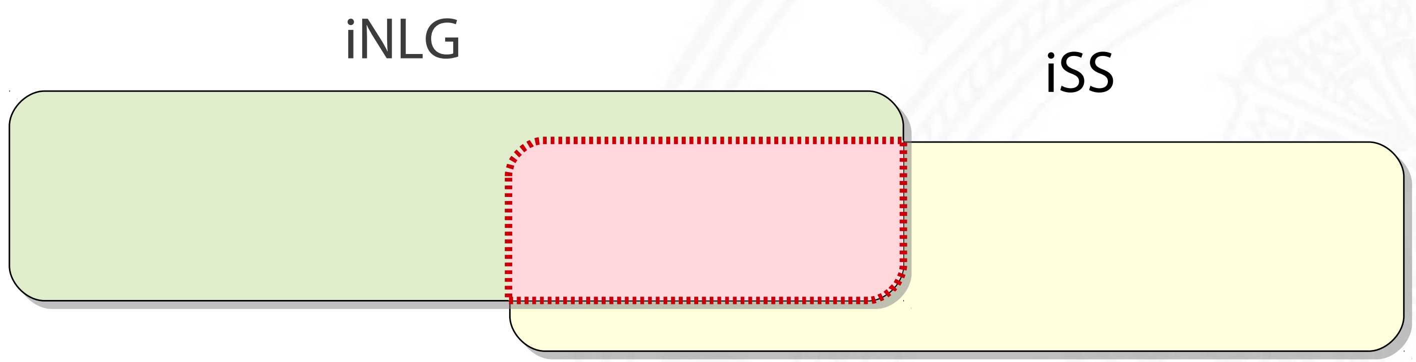
Outline of the Talk

✓ Goals for Incremental Speech Output

- Incremental Speech Output
  - Incremental Natural Language Generation (iNLG)
  - Incremental Speech Synthesis (iSS)
- Application & Results:
  - Massively Reduced System Latency
  - Adaptive Information Presentation Preferred by Listeners

Incremental Speech Output: Overview

- split up into two (generic) processors:
  - natural language generation (iNLG)
  - speech synthesis (iSS)

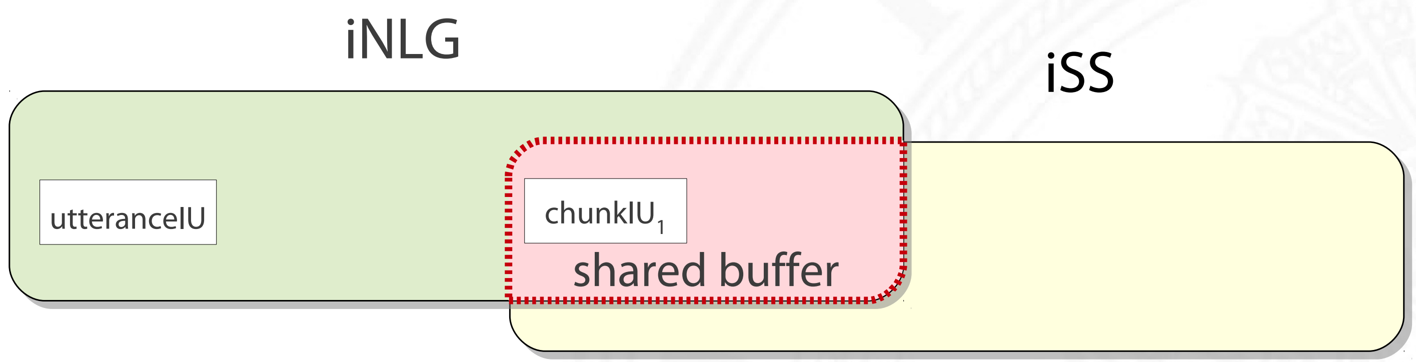


- implemented in the IU framework using InproTK
  - available as open-source: http://inprotk.sourceforge.net

(Schlangen and Skantze, 2009; Baumann and Schlangen, 2012)

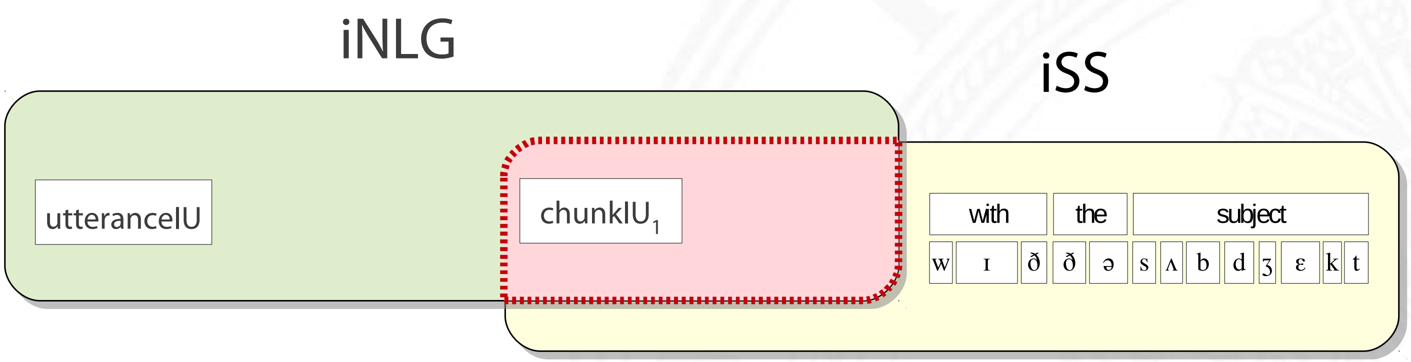
Incremental Speech Output: Overview

- starting with an utterance description
- iNLG splits the utterance in chunks and outputs one chunk to the buffer that is shared with iSS



Incremental Speech Output: Overview

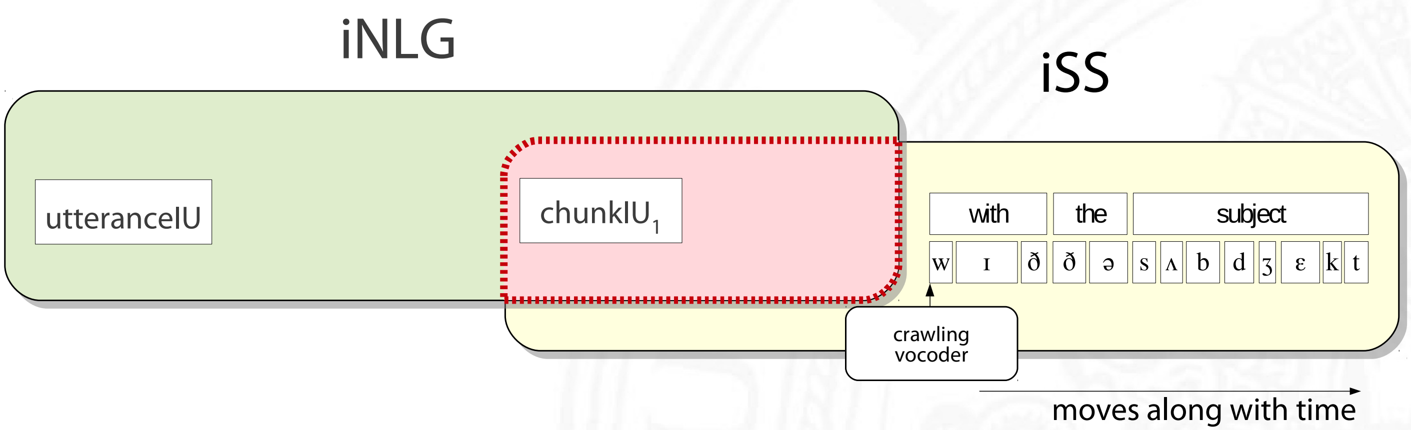
- iSS processes chunk to produce phonemes



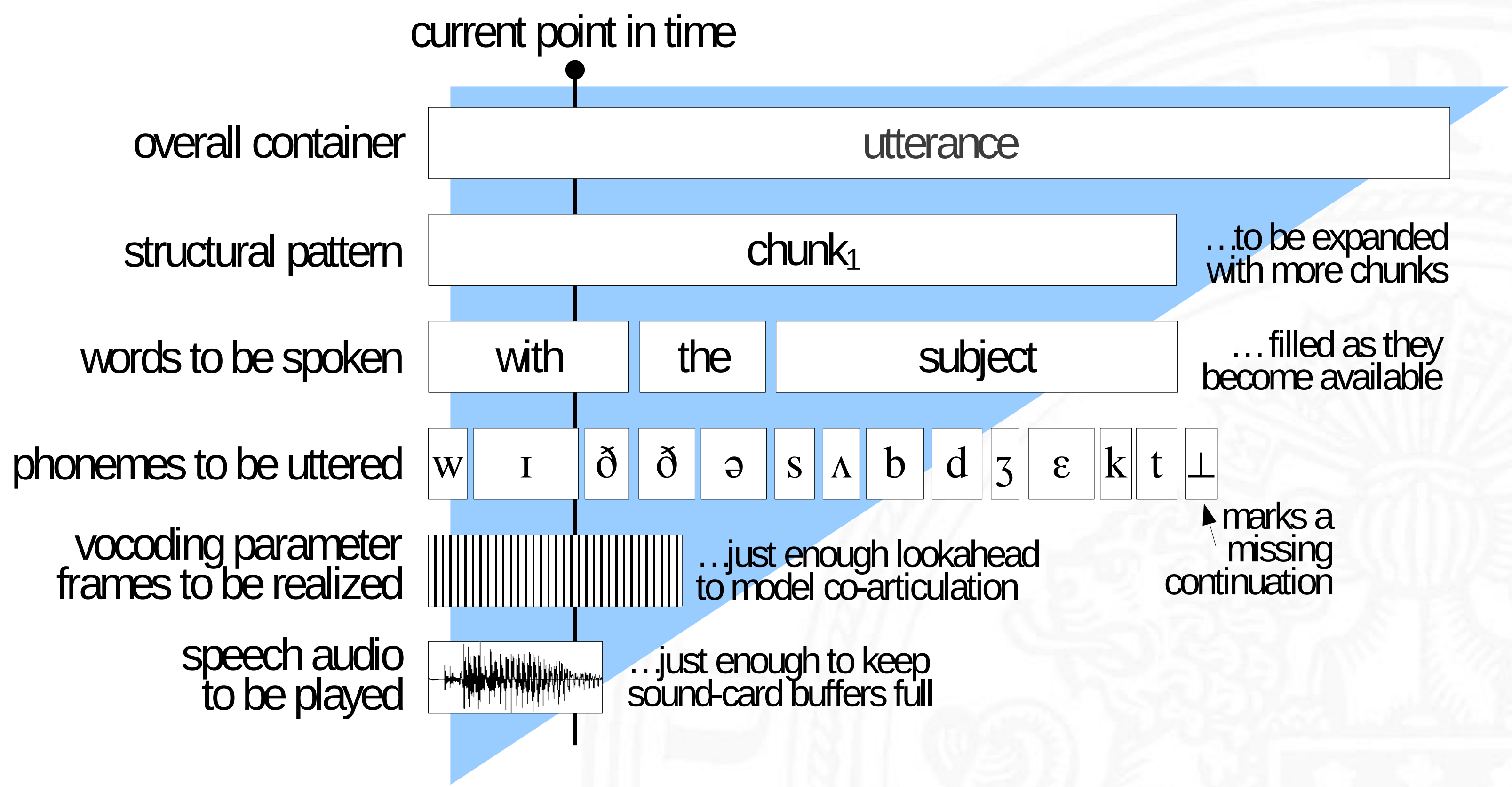
linguistic processing via MaryTTS (Schröder & Trouvain, 2003)

Incremental Speech Output: Overview

- iSS processes chunk and
- synthesizes *just-in-time*
  (only with enough look-ahead to keep all buffers full)



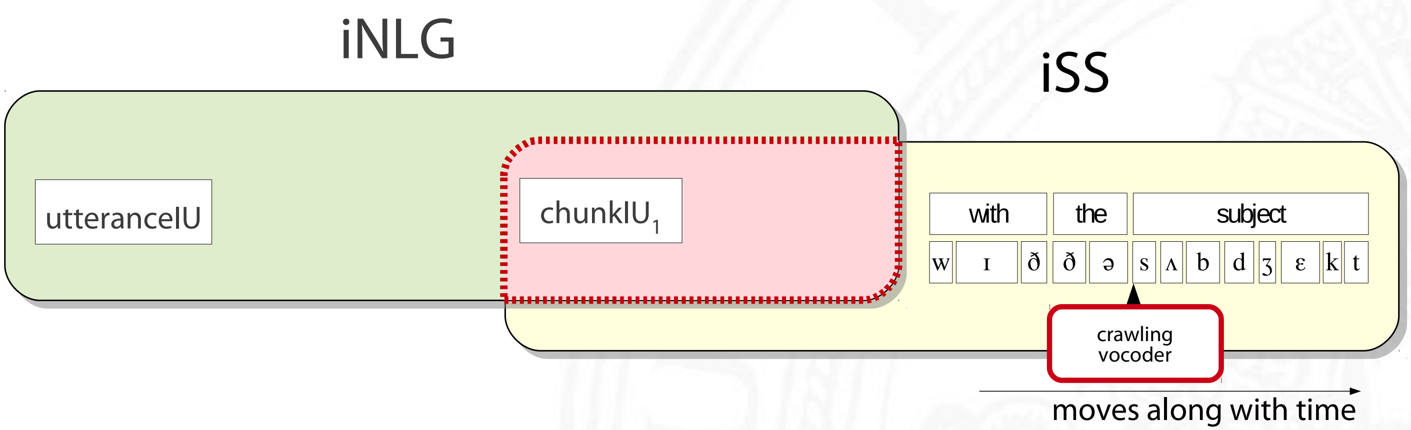
a *Just-In-Time* Formulation for Incremental Speech Synthesis



- triangular „top-down-left-to-right“ data structure

Incremental Speech Output: Overview

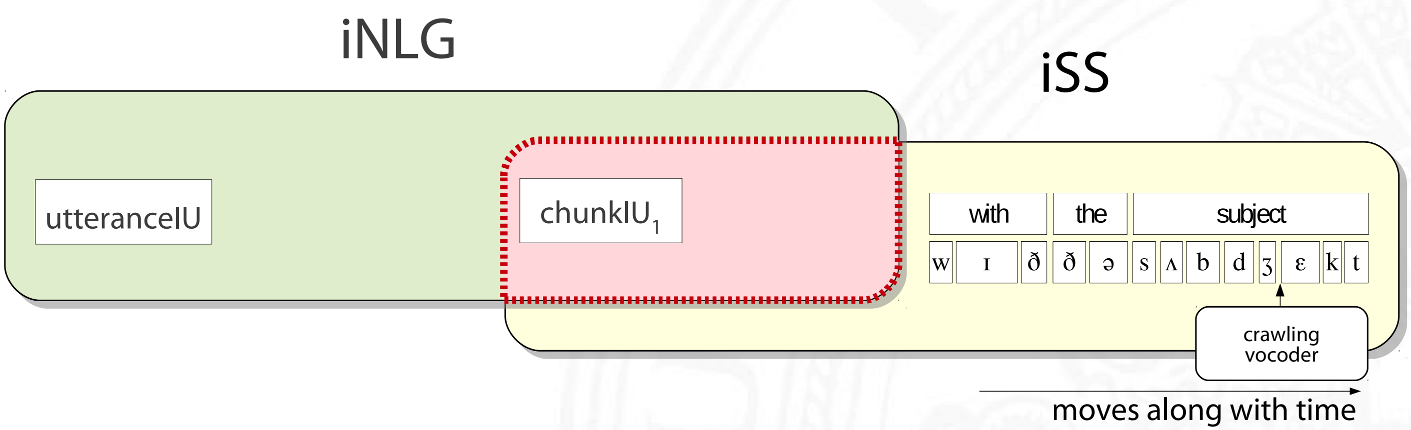
- using a *crawling vocoder* that performs HMM optimization and vocoding in real-time



(largely based on MaryTTS code; see also Dutoit et al., 2011)

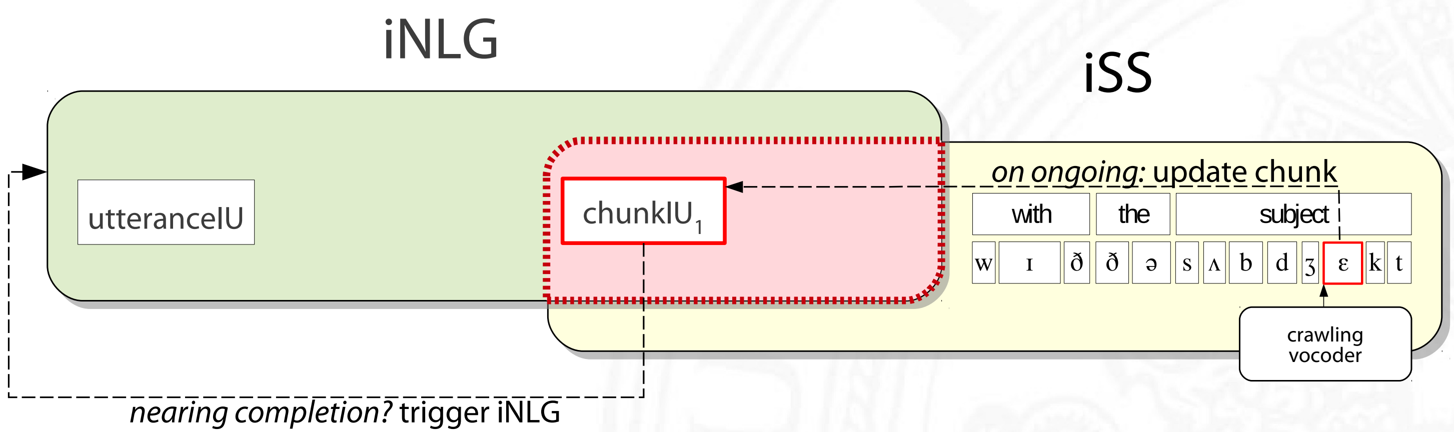
Incremental Speech Output: Overview

- using a *crawling vocoder* that performs HMM optimization and vocoding in real-time
- when nearing the end of the current chunk …



Incremental Speech Output: Overview

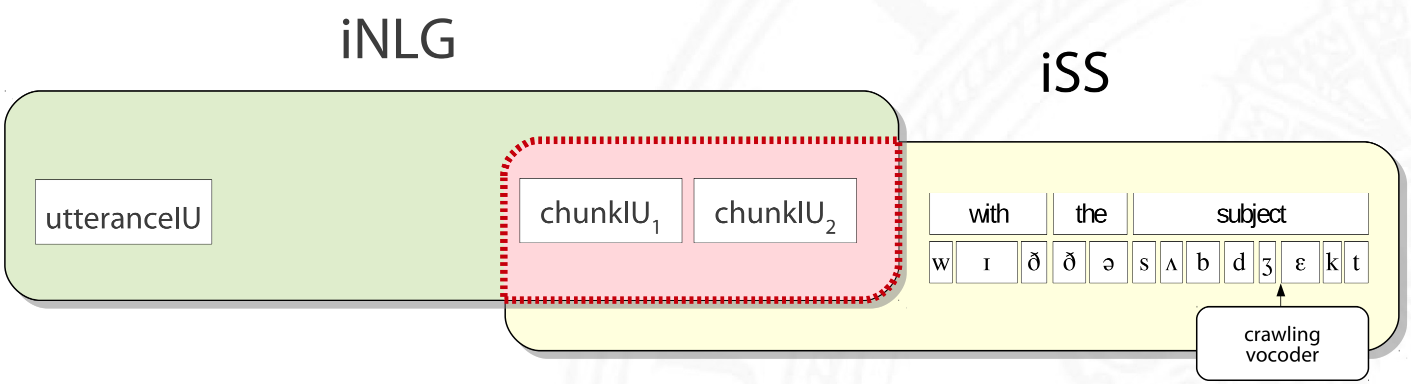
- update-messages are sent from phonemes to chunk to iNLG



(this is a generic update mechanism in INPROTK)

Incremental Speech Output: Overview

- and iNLG adds another chunkIU before synthesis runs out of speech
- it's integrated & appended to the ongoing synthesis



- the process repeats until all chunks are synthesized

Incremental Speech Output: Summary

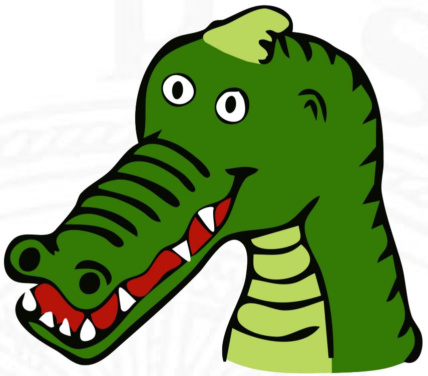
- two components:
  - iNLG: turns ideas into words
  - iSS: turns words into speech audio
- features:
  - low-latency changes to upcoming chunks
  - highly modular implementation of the components
- questions:
  - what exactly are these chunks?
  - how can we ensure utterance cohesion?
  - what's the chunks' granularity?

Granularity of Incremental Chunks

- granularity ≙ size of the units
  - determines responsiveness to changes
  - determines context available for coherent processing
- ideally: generate word-by-word
  - ➔ highly responsive behaviour

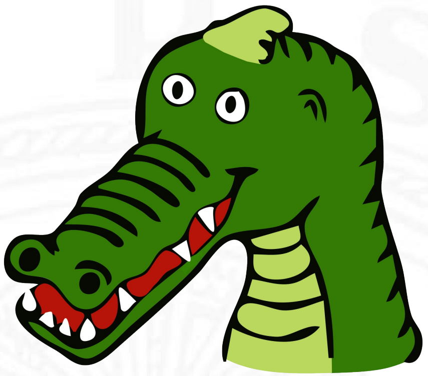
Granularity of Incremental Chunks for Language Generation

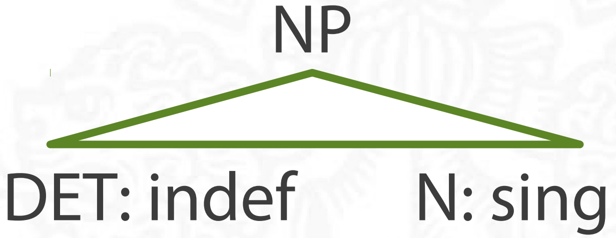
- granularity ≙ size of the units
  - determines responsiveness to changes
  - determines context available
- ideally: generate word-by-word
  - however, this may be infeasible



Granularity of Incremental Chunks for Language Generation

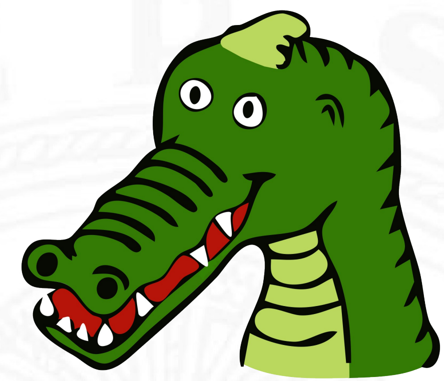
- granularity ≙ size of the units
  - determines responsiveness to changes
  - determines context available
- ideally: generate word-by-word
  - however, this may be infeasible

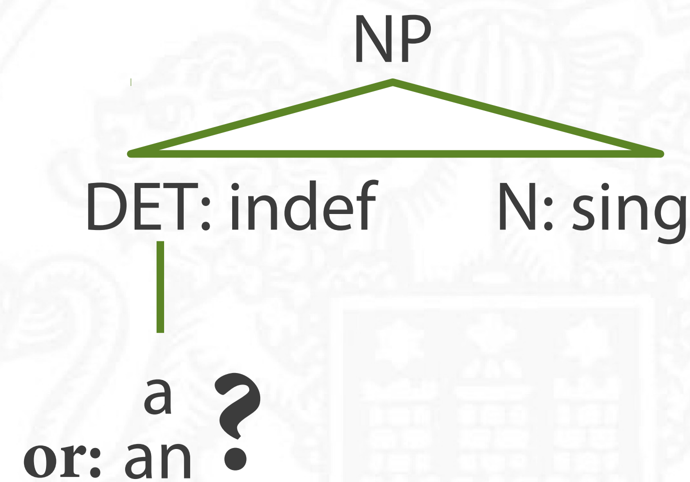




Granularity of Incremental Chunks for Language Generation

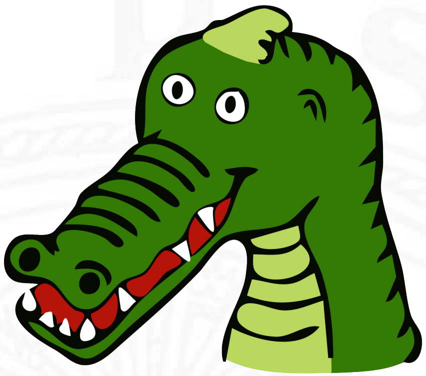
- granularity ≙ size of the units
  - determines responsiveness to changes
  - determines context available
- ideally: generate word-by-word
  - however, this may be infeasible

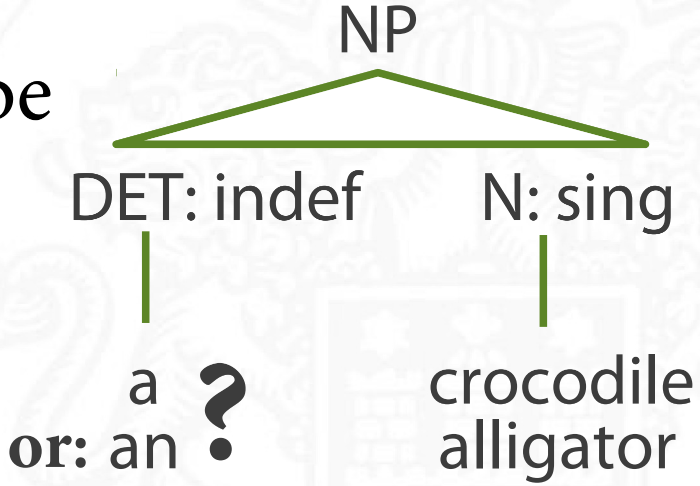




Granularity of Incremental Chunks for Language Generation

- granularity ≙ size of the units
  - determines responsiveness to changes
  - determines context available
- ideally: generate word-by-word
  - however, this may be infeasible
- surface structure cannot always be produced purely left-to-right and word-by-word





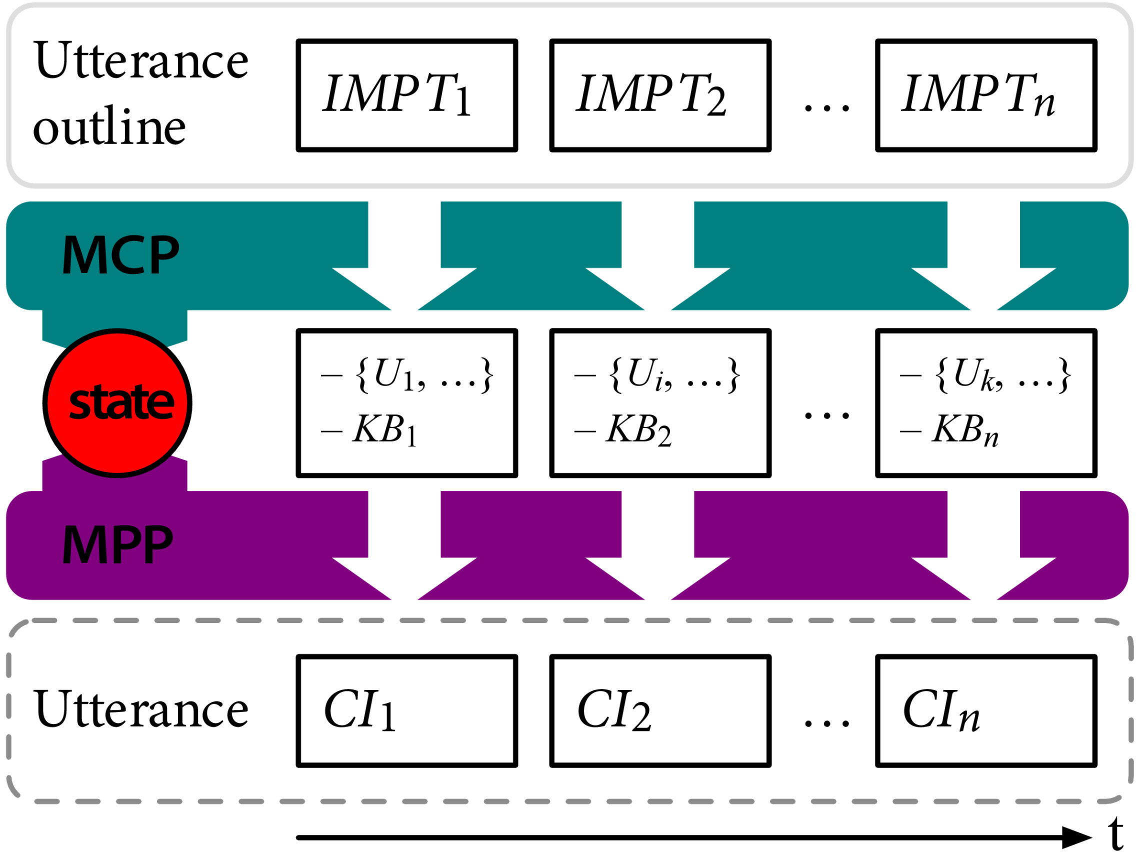
Granularity of Incremental Chunks for Speech Synthesis

- input units should ensure a coherent prosodic realization
  - „This. must. be. avoided.“
  - allow for some lookahead into the future

➔ our sub-utterance chunks:
  - roughly correspond to intonation phrases
  - coarse granularity of incremental generation
  - *ideal* size remains an open research question

Incremental Natural Language Generation

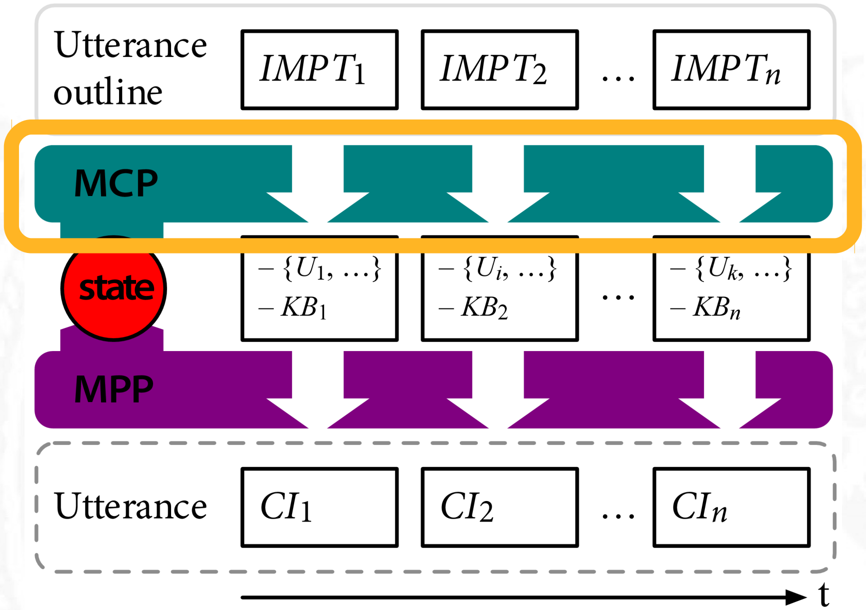
- we combine two interacting sub-components that share a common state



(please ask Hendrik Buschmeier for details)

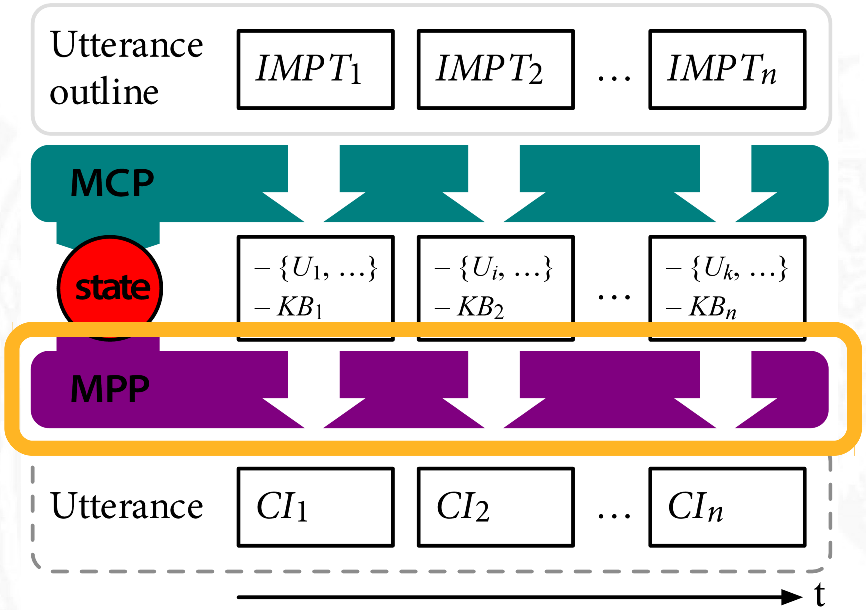
Micro-Content Planning (MCP)

- turns utterance outline into
  - set of desired updates on listener's information state
  - presuppositions and private knowledge
- generates incremental micro-planning tasks (IMPTs, one at a time)



Micro-Planning Proper (MPP)

- takes one IMPT
- uses SPUD to generate surface form
- adds generated communicative intent to **common state** between MCP and MPP
  - taken into account for generation of next IMPT
  - for coherence & adherence to pragmatic principles



our implementation uses JavaSPUD (DeVault, 2008)

Combining Incremental Language Generation ✓
and Incremental Speech Synthesis ✓
for **Adaptive Information Presentation**

Example Application: Reading out Calendar Events

- part of a virtual human systems project
- relatively long utterances:
    - example: play ReferenceExample1.aiff
      „your appointment on Wednesday, 4. April, 10 am to 12 pm, titled Lecture Linguistics has been rescheduled to Friday, 6. April, 12 pm to 2 pm.“
- 6-7 chunks of information

Advantage of iNLG+iSS: Processing Time

- system response time (i.e. processing until audio onset) is crucial in an interactive environment
- a non-incremental system must perform all processing utterance-initially
- an incremental system can *fold* most processing time into delivery time

Results for Utterance Onset Timing

processing step	non-incr.	incremental
NLG	361	52
Synth. (ling. processing)	217	222
Synth. (HMM & vocoding)	1004	21
Total	1582	**295**

averaged over 9 stimuli, time in milliseconds

- iNLG and iSS can start output much faster than non-incremental processing
- (linguistic pre-processing is not yet incrementalized)

Evaluation of Adaptive Behaviour

- lowest hanging fruit: deal with intermittent noise (e.g. to be able to use this next to a busy street)
  - at random intervals, noise is played
- simple behaviours to cope with noise:
  - ignore the noise, continue speaking (baseline A)
  - stop audio, continue after end of noise (baseline B)
- example: play {A,B}5.aiff

Adaptation Strategies

1. „high-level": repetition (of selected chunks)
2. prosodic adaptation to noise
3. incremental NLG allows for dynamic, adapted creation of later sub-utterance chunks
   - adaptation to state happens in both MCP and MPP:

- MCP
  - which IMPT next?
  - repair/comment?
- MPP
  - influence generation parameters, such as verbosity, redundancy

Application: Adaptive Behaviour

- simple behaviours to cope with noise:
  - ignore the noise, continue speaking (baseline A)
  - stop audio, continue after end of noise (baseline B)
- adaptive behaviour:
  - stop delivery at the end of current word,
  - restart adapted phrase after noise (iNLG+iSS)
- example: play C5.aiff

User Study

- 9 stimuli × 3 conditions (A, B, iNLG+iSS)
- Q: „I found the behaviour of the system in this situation as I would expect it from a human speaker"
- 12 subjects, 7-point Likert scale

→ highly significant preference for incremental system

→ no difference between settings A and B

  → stopping audio did not improve user ratings !!

Conclusion

- we present a method for incremental NLG
- we present a system for incremental speech synthesis
    - just-in-time, low-latency, low overhead for changes
    - general purpose, open-source
- show performance in interactive environment
    - radically reduced system onset time
    - adaptation to intermittent noise
    - highly preferred by human listeners

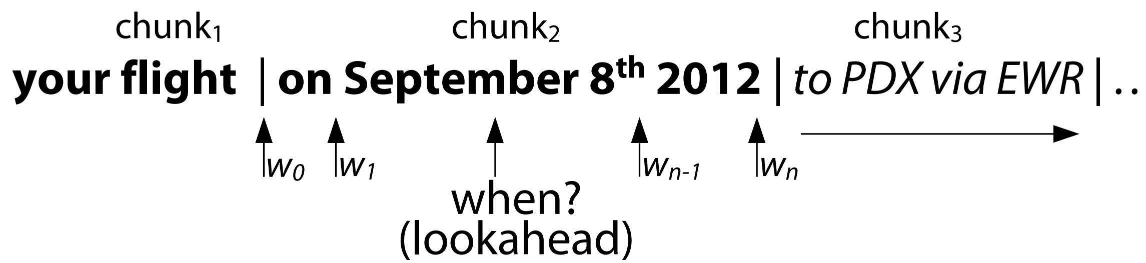
Future Work

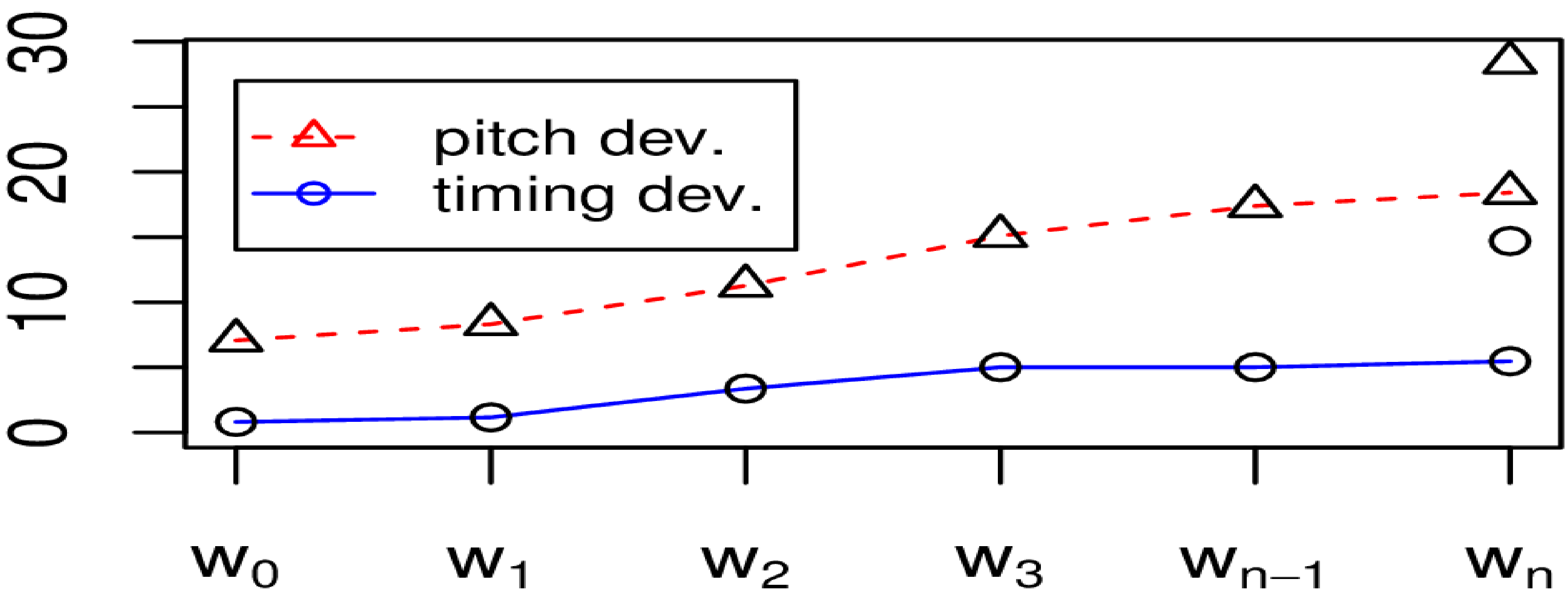
- research question: *ideal* granularity for NLG and iSS
- further develop mid-level incremental structure & processing for improved prosody production
  - also incrementalize the HMM state selection (which currently uses decision tree features that look into the future – however, is this necessary?)
- extend system to handle intra-utterance user feedback, interruptions, …

Thank you !
Questions and Comments ?

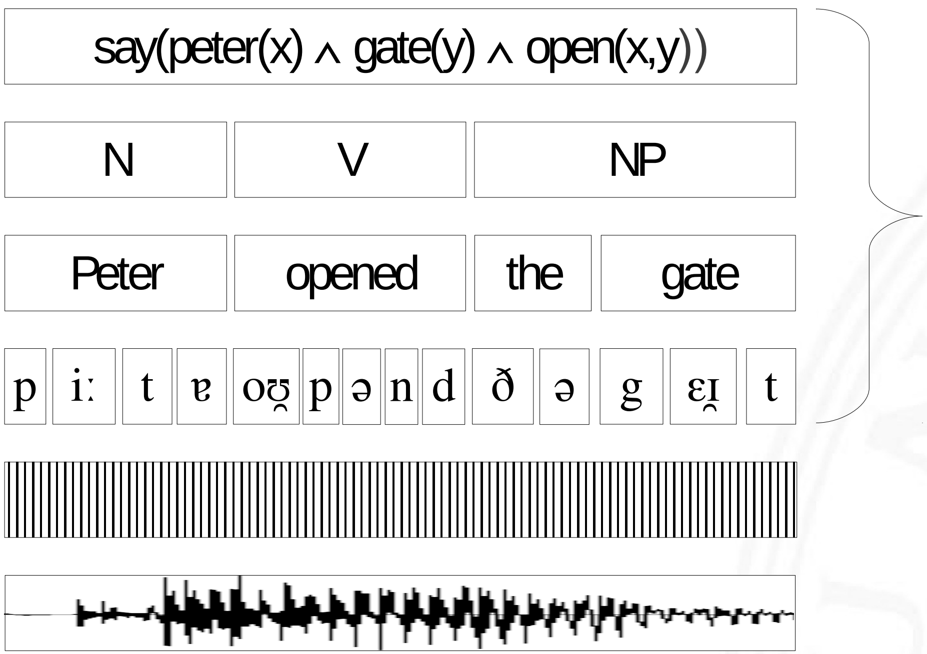
Thank you very much for your attention.

Prosodic Quality of Incremental Speech Synthesis





Advantages of iSS: Computational Costs



rough estimates for MaryTTS

symbolic processing → cheap	**20 %**
large set of linear equations	**40 %**
lots of signal processing	**40 %**

Speech Synthesis is fast, why not re-do it repeatedly?

- it may be fast on the server,
  but it's still slow on your phone
  - repeating drains the battery more than necessary
- you need a notion of how to align the old and the new synthesis – that's at least as difficult as what we're doing

Adaptation Used in the System

- Re-synthesis in new context results in utterance-initial prosody
- Details on NLG adaptation in the paper