# INPRO_iSS: A Component for Just-In-Time Incremental Speech Synthesis

**Timo Baumann**
University of Hamburg
Department for Informatics
Germany
baumann@informatik.uni-hamburg.de

**David Schlangen**
University of Bielefeld
Faculty of Linguistics and Literary Studies
Germany
david.schlangen@uni-bielefeld.de

## Abstract

We present a component for incremental speech synthesis (iSS) and a set of applications that demonstrate its capabilities. This component can be used to increase the responsivity and naturalness of spoken interactive systems. While iSS can show its full strength in systems that generate output incrementally, we also discuss how even otherwise unchanged systems may profit from its capabilities.

## 1 Introduction

Current state of the art in speech synthesis for spoken dialogue systems (SDSs) is for the synthesis component to expect full utterances (in textual form) as input and to deliver an audio stream verbalising this full utterance. At best, timing information is returned as well so that a control component can determine in case of an interruption / barge-in by the user where in the utterance this happened (Edlund, 2008; Matsuyama et al., 2010).

We want to argue here that providing capabilities to speech synthesis components for dealing with units smaller than full utterances can be beneficial for a whole range of interactive speech-based systems. In the easiest case, incremental synthesis simply reduces the utterance-initial delay before speech output starts, as output already starts when its beginning has been produced. In an otherwise conventional dialogue system, the synthesis module could make it possible to interrupt the output speech stream (e. g., when a noise event is detected that makes it likely that the user will not be able to hear what is being said), and continue production when the interruption is over. If other SDS components are adapted more to take advantage of incremental speech synthesis, even more flexible behaviours can be realised, such as providing utterances in *installments* (Clark, 1996) that prompt for backchannel signals, which in turn can prompt different utterance continuations, or starting an utterance before all information required in the utterance is available ("so, uhm, there are flights to Seoul on uh ..."), signaling that the turn is being held. Another, less conventional type of speech-based system that could profit from iSS is "babelfish-like" simultaneous speech-to-speech translation.

Research on architectures, higher-level processing modules and lower-level processing modules that would enable such behaviour is currently underway (Skantze and Schlangen, 2009; Skantze and Hjalmarsson, 2010; Baumann and Schlangen, 2011), but a synthesis component that would unlock the full potential of such strategies is so far missing. In this paper, we present such a component, which is capable of

(a) starting to speak before utterance processing has finished;
(b) handling edits made to (as-yet unspoken) parts of the utterance even while a prefix is already being spoken;
(c) enabling adaptations of delivery parameters such as speaking rate or pitch;
(d) autonomously making appropriate delivery-related decisions;
(e) providing information about progress in delivery; and, last but not least,
(f) running in real time.

Our iSS component is built on top of an existing non-incremental synthesis component, MaryTTS (Schröder and Trouvain, 2003), and on an existing architecture for incremental processing, INPROTK (Baumann and Schlangen, 2012).



*Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 103–108,
Jeju, Republic of Korea, 8-14 July 2012. ©2012 Association for Computational Linguistics

After a discussion of related work (Section 2), we describe the basic elements of our iSS component (Section 3) and some demonstrator applications that we created which showcase certain abilities.[1]

## 2 Related Work

Typically, in current SDSs utterances are generated (either by lookup/template-based generation, or, less commonly, by concept-to-utterance natural language generation (NLG)) and then synthesised in full (McTear, 2002). There is very little work on incremental synthesis (i.e., one that would work with units smaller than full utterances). Edlund (2008) outlines some requirements for incremental speech synthesis: to *give constant feedback* to the dialogue system about what has been delivered, to *be interruptible* (and possibly continue from that position), and to *run in real time*. Edlund (2008) also presents a prototype that meets these requirements, but is limited to diphone synthesis that is performed non-incrementally before utterance delivery starts. We go beyond this in processing just-in-time, and also enabling changes during delivery.

Skantze and Hjalmarsson (2010) describe a system that generates utterances incrementally (albeit in a WOz-enviroment), allowing earlier components to incrementally produce and revise their hypothesis about the user's utterance. The system can automatically play hesitations if by the time it has the turn it does not know what to produce yet. They show that users prefer such a system over a non-incremental one, even though it produced longer dialogues. Our approach is complementary to this work, as it targets a lower layer, the realisation or synthesis layer. Where their system relies on 'regular' speech synthesis which is called on relatively short utterance fragments (and thus pays for the increase in responsiveness with a reduction in synthesis quality, esp. regarding prosody), we aim to incrementalize the speech synthesis component itself.

Dutoit et al. (2011) have presented an incremental formulation for HMM-based speech synthesis. However, their system works offline and is fed by non-incrementally produced phoneme target sequences. We aim for a fully incremental speech synthesis component that can be integrated into dialogue systems.

There is some work on incremental NLG (Kilger and Finkler, 1995; Finkler, 1997; Guhe, 2007); however, that work does not concern itself with the actual synthesis of speech and hence describes only what would generate the input to our component.

## 3 Incremental Speech Synthesis

### 3.1 Background on Speech Synthesis

Text-to-speech (TTS) synthesis normally proceeds in a top-down fashion, starting on the utterance level (for stress patterns and sentence-level intonation) and descending to words and phonemes (for pronunciation details), in order to make globally optimised decisions (Taylor, 2009). In that way, target phoneme sequences annotated with durations and pitch contours are generated, in what is called the linguistic pre-processing step.

The then following synthesis step proper can be executed in one of several ways, with HMM-based and unit-selection synthesis currently being seen as producing the perceptually best results (Taylor, 2009). The former works by first turning the target sequence into a sequence of HMM states; a global optimization then computes a stream of vocoding features that optimize both HMM emission probabilities and continuity constraints (Tokuda et al., 2000). Finally, the parameter frames are fed to a vocoder which generates the speech audio signal. Unit-selection, in contrast, searches for the best sequence of (variably sized) units of speech in a large, annotated corpus of recordings, aiming to find a sequence that closely matches the target sequence.

As mentioned above, Dutoit et al. (2011) have presented an online formulation of the optimization step in HMM-based synthesis. Beyond this, two other factors influenced our decision to follow the HMM-based approach: (a) HMM-based synthesis nicely separates the production of vocoding parameter frames from the production of the speech audio signal, which allows for more fine-grained concurrent processing (see next subsection); (b) parameters are partially independent in the vocoding frames, which makes it possible to manipulate e. g. pitch independently (and outside of the HMM framework) without altering other parameters or deteriorating speech quality.

[1] The code of the toolkit and its iSS component and the demo applications discussed below have been released as open-source at `http://inprotk.sourceforge.net`.

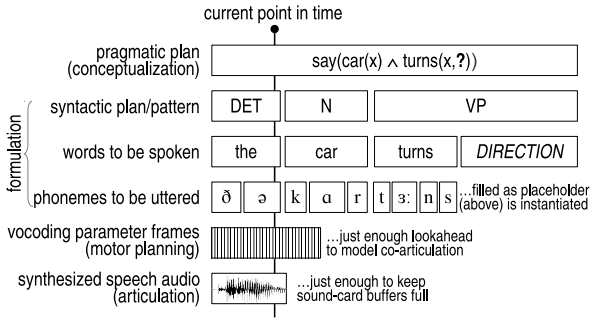


Figure 1: Hierarchic structure of incremental units describing an example utterance as it is being produced during utterance delivery.

### 3.2 System Architecture

Our component works by reducing the aforementioned top-down requirements. We found that it is not necessary to work out all details at one level of processing before starting to process at the next lower level. For example, not all *words* of the utterance need to be known to produce the sentence-level intonation (which itself however is necessary to determine pitch contours) as long as a structural outline of the utterance is available. Likewise, post-lexical phonological processes can be computed as long as a local context of one word is available; vocoding parameter computation (which must model co-articulation effects) in turn can be satisfied with just one phoneme of context; vocoding itself does not need any lookahead at all (aside from audio buffering considerations).

Thus, our component generates its data structures incrementally in a top-down-and-left-to-right fashion with different amounts of pre-planning, using several processing modules that work concurrently. This results in a 'triangular' structure (illustrated in Figure 1) where only the absolutely required minimum has to be specified at each level, allowing for later adaptations with few or no recomputations required.

As an aside, we observe that our component's architecture happens to correspond rather closely to Levelt's (1989) model of human speech production. Levelt distinguishes several, partially independent processing modules (conceptualization, formulation, articulation, see Figure 1) that function incrementally and "in a highly automatic, reflex-like way" (Levelt, 1989, p. 2).

### 3.3 Technical Overview of Our System

As a basis, we use MaryTTS (Schröder and Trouvain, 2003), but we replace Mary's internal data structures with structures that support incremental specifications; these we take from an extant incremental spoken dialogue system architecture and toolkit, INPROTK (Schlangen et al., 2010; Baumann and Schlangen, 2012). In this architecture, incremental processing as the processing of *incremental units* (IUs), which are the smallest 'chunks' of information at a specific level (such as words, or phonemes, as can be seen in Figure 1). IUs are interconnected to form a network (e. g. words keep links to their associated phonemes, and vice-versa) which stores the system's complete information state.

The iSS component takes an IU sequence of chunks of words as input (from an NLG component). Crucially, this sequence can then still be modified, through: (a) *continuations*, which simply link further words to the end of the sequence; or (b) *replacements*, where elements in the sequence are "unlinked" and other elements are spliced in. Additionally, a chunk can be marked as *open*; this has the effect of linking to a special *hesitation word*, which is produced only if it is not replaced (by the NLG) in time with other material.

Technically, the representation levels below the chunk level are generated in our component by MaryTTS's linguistic preprocessing and converting the output to IU structures. Our component provides for two modes of operation: Either using MaryTTS' HMM optimization routines which non-incrementally solve a large matrix operation and subsequently iteratively optimize the global variance constraint (Toda and Tokuda, 2007). Or, using the incremental algorithm as proposed by Dutoit et al. (2011). In our implementation of this algorithm, HMM emissions are computed with one phoneme of context in both directions; Dutoit et al. (2011) have found this setting to only slightly degrade synthesis quality. While the former mode incurs some utterance-initial delay, switching between alternatives and prosodic alteration can be performed at virtually no lookahead, while requiring just little lookahead for the truly incremental mode. The resulting vocoding frames then are attached to their corresponding phoneme units. Phoneme units then contain all the information

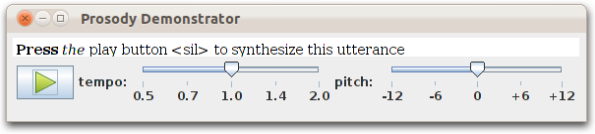


Figure 2: Example application that showcases just-in-time manipulation of prosodic aspects (tempo and pitch) of the ongoing utterance.

needed for the final vocoding step, in an accessible form, which makes possible various manipulations before the final synthesis step.

The lowest level module of our component is what may be called a *crawling vocoder*, which actively moves along the phoneme IU layer, querying each phoneme for its parameter frames one-by-one and producing the corresponding audio via vocoding. The vocoding algorithm is entirely incremental, making it possible to vocode "just-in-time": only when audio is needed to keep the sound card buffer full does the vocoder query for a next parameter frame. This is what gives the higher levels the maximal amount of time for re-planning, i. e., to be incremental.

### 3.4 Quality of Results

As these descriptions should have made clear, there are some elements in the processing steps in our iSS component that aren't yet fully incremental, such as assigning a sentence-level prosody. The best results are thus achieved if a full utterance is presented to the component initially, which is used for computation of prosody, and of which then elements may be changed (e. g., adjectives are replaced by different ones) on the fly. It is unavoidable, though, that there can be some "breaks" at the seams where elements are replaced. Moreover, the way feature frames can be modified (as described below) and the incremental HMM optimization method may lead to deviations from the global optimum. Finally, our system still relies on Mary's non-incremental HMM state selection technique which uses decision trees with non-incremental features.

However, preliminary evaluation of the component's prosody given varying amounts of lookahead indicate that degradations are reasonably small. Also, the benefits in naturalness of behaviour enabled by iSS may outweigh the drawback in prosodic quality.

## 4 Interface Demonstrations

We will describe the features of iSS, their implementation, their programming interface, and corresponding demo applications in the following subsections.

### 4.1 Low-Latency Changes to Prosody

Pitch and tempo can be adapted on the phoneme IU layer (see Figure 1). Figure 2 shows a demo interface to this functionality. Pitch is determined by a single parameter in the vocoding frames and can be adapted independently of other parameters in the HMM approach. We have implemented capabilities of adjusting all pitch values in a phoneme by an offset, or to change the values gradually for all frames in the phoneme. (The first feature is show-cased in the application in Figure 2, the latter is used to cancel utterance-final pitch changes when a continuation is appended to an ongoing utterance.) Tempo can be adapted by changing the phoneme units' durations which will then repeat (or skip) parameter frames (for lengthened or shortened phonemes, respectively) when passing them to the *crawling vocoder*. Adaptations are conducted with virtually no lookahead, that is, they can be executed even on a phoneme that is currently being output.

### 4.2 Feedback on Delivery

We implemented a fine-grained, hierarchical mechanism to give detailed feedback on delivery. A new *progress* field on IUs marks whether the IU's production is upcoming, ongoing, or completed. Listeners may subscribe to be notified about such progress changes using an update interface on IUs. The applications in Figures 2 and 4 make use of this interface to mark the words of the utterance in bold for completed, and in italic for ongoing words (incidentally, the screenshot in Figure 4 was taken exactly at the boundary between "delete" and "the").

### 4.3 Low-Latency Switching of Alternatives

A major goal of iSS is to change what is being said while the utterance is ongoing. Forward-pointing same-level links (SLLs, (Schlangen and Skantze, 2009; Baumann and Schlangen, 2012)) as shown in Figure 3 allow to construct alternative utterance paths beforehand. Deciding on the actual utterance continuation is a simple re-ranking of the forward

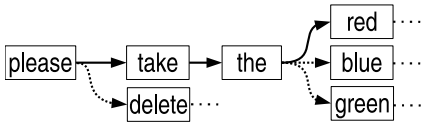


Figure 3: Incremental units chained together via forward-pointing same-level links to form an utterance tree.

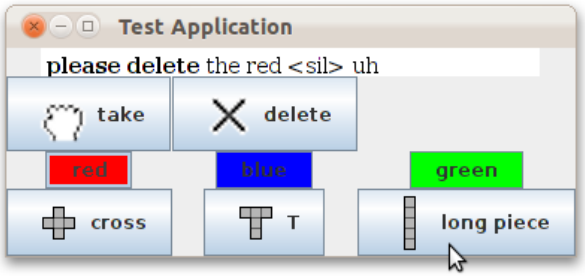


Figure 4: Example application to showcase just-in-time selection between different paths in a complex utterance.

SLLs which can be changed until immediately before the word (or phoneme) in question is being uttered.

The demo application shown in Figure 4 allows the user to select the path through a fairly complex utterance tree. The user has already decided on the color, but not on the type of piece to be deleted and hence the currently selected plan is to play a hesitation (see below).

### 4.4 Extension of the Ongoing Utterance

In the previous subsection we have shown how alternatives in utterances can be selected with very low latency. Adding continuations (or alternatives) to an ongoing utterance incurs some delay (some hundred milliseconds), as we ensure that an appropriate sentence-level prosody for the alternative (or continuation) is produced by re-running the linguistic pre-processing on the complete utterance; we then integrate only the new, changed parts into the IU structure (or, if there still is time, parts just before the change, to account for co-articulation).

Thus, practical applications which use incremental NLG must generate their next steps with some lookahead to avoid stalling the output. However, utterances can be marked as non-final, which results in a special hesitation word being inserted, as explained below.

### 4.5 Autonomously Performing Disfluencies

In a multi-threaded, real-time system, the *crawling vocoder* may reach the end of synthesis before the NLG component (in its own thread) has been able to add a continuation to the ongoing utterance. To avoid this case, special *hesitation word*s can be inserted at the end of a yet unfinished utterance. If the crawling vocoder nears such a word, a hesitation will be played, unless a continuation is available. In that case, the hesitation is skipped (or aborted if currently ongoing).[2]

### 4.6 Type-to-Speech

A final demo application show-cases truly incremental HMM synthesis taken to its most extreme: A text input window is presented, and each word that is typed is treated as a single-word chunk which is immediately sent to the incremental synthesizer. (For this demonstration, synthesis is slowed to half the regular speed, to account for slow typing speeds and to highlight the prosodic improvements when more right context becomes available to iSS.) A use case with a similar (but probably lower) level of incrementality could be simultaneous speech-to-speech translation, or type-to-speech for people with speech disabilities.

## 5 Conclusions

We have presented a component for incremental speech synthesis (iSS) and demonstrated its capabilities with a number of example applications. This component can be used to increase the responsivity and naturalness of spoken interactive systems. While iSS can show its full strengths in systems that also generate output incrementally (a strategy which is currently seeing some renewed attention), we discussed how even otherwise unchanged systems may profit from its capabilities, e. g., in the presence of intermittent noise. We provide this component in the hope that it will help spur research on incremental natural language generation and more interactive spoken dialogue systems, which so far had to made do with inadequate ways of realising its output.

[2]Thus, in contrast to (Skantze and Hjalmarsson, 2010), hesitations do not take up any additional time.

## References


Timo Baumann and David Schlangen. 2011. Predicting the Micro-Timing of User Input for an Incremental Spoken Dialogue System that Completes a User's Ongoing Turn. In *Proceedings of SigDial 2011*, pages 120–129, Portland, USA, June.

Timo Baumann and David Schlangen. 2012. The INPROTK 2012 release. In *Proceedings of SDCTD*. to appear.

Herbert H. Clark. 1996. *Using Language*. Cambridge University Press.

Thierry Dutoit, Maria Astrinaki, Onur Babacan, Nicolas d'Alessandro, and Benjamin Picart. 2011. pHTS for Max/MSP: A Streaming Architecture for Statistical Parametric Speech Synthesis. Technical Report 1, numediart Research Program on Digital Art Technologies, March.

Jens Edlund. 2008. Incremental speech synthesis. In *Second Swedish Language Technology Conference*, pages 53–54, Stockholm, Sweden, November. System Demonstration.

Wolfgang Finkler. 1997. *Automatische Selbstkorrektur bei der inkrementellen Generierung gesprochener Sprache unter Realzeitbedingungen*. Dissertationen zur Künstlichen Intelligenz. infix Verlag.

Markus Guhe. 2007. *Incremental Conceptualization for Language Production*. Lawrence Erlbaum Asso., Inc., Mahwah, USA.

Anne Kilger and Wolfgang Finkler. 1995. Incremental Generation for Real-time Applications. Technical Report RR-95-11, DFKI, Saarbrücken, Germany.

William J.M. Levelt. 1989. *Speaking: From Intention to Articulation*. MIT Press.

Kyoko Matsuyama, Kazunori Komatani, Ryu Takeda, Toru Takahashi, Tetsuya Ogata, and Hiroshi G. Okuno. 2010. Analyzing User Utterances in Barge-in-able Spoken Dialogue System for Improving Identification Accuracy. In *Proceedings of Interspeech*, pages 3050–3053, Makuhari, Japan, September.

Michael McTear. 2002. *Spoken Dialogue Technology. Toward the Conversational User-Interface*. Springer, London, UK.

David Schlangen and Gabriel Skantze. 2009. A General, Abstract Model of Incremental Dialogue Processing. In *Proceedings of the EACL*, Athens, Greece.

David Schlangen, Timo Baumann, Hendrik Buschmeier, Okko Buß, Stefan Kopp, Gabriel Skantze, and Ramin Yaghoubzadeh. 2010. Middleware for Incremental Processing in Conversational Agents. In *Proceedings of SigDial 2010*, pages 51–54, Tokyo, Japan, September.

Marc Schröder and Jürgen Trouvain. 2003. The German Text-to-Speech Synthesis System MARY: A Tool for Research, Development and Teaching. *International Journal of Speech Technology*, 6(3):365–377, October.

Gabriel Skantze and Anna Hjalmarsson. 2010. Towards incremental speech generation in dialogue systems. In *Proceedings of SigDial 2010*, pages 1–8, Tokyo, Japan, September.

Gabriel Skantze and David Schlangen. 2009. Incremental dialogue processing in a micro-domain. In *Proceedings of EACL 2009*, Athens, Greece, April.

Paul Taylor. 2009. *Text-to-Speech Synthesis*. Cambridge Univ Press, Cambridge, UK.

Tomoki Toda and Keiichi Tokuda. 2007. A Speech Parameter Generation Algorithm Considering Global Variance for HMM-based Speech Synthesis. *IEICE Transactions on Information and Systems*, 90(5):816–824.

Keiichi Tokuda, Takayoshi Yoshimura, Takashi Masuko, Takao Kobayashi, and Tadashi Kitamura. 2000. Speech Parameter Generation Algorithms for HMM-based Speech Synthesis. In *Proceedings of ICASSP 2000*, pages 1315–1318, Istanbul, Turkey.