

# Japec

## A JAPE-to-Java Optimizing Compiler

(c) Copyright 2005-2006 Ontotext Lab, Sirma Group Corp.

---

3 Mar, 2006

# JAPE and Jappec Overview

---

- **JAPE is a language** which provides finite-state transduction over annotations based on regular expressions
  - JAPE is “the language of GATE”
  - GATE is a leading NLP/text engineering platform, <http://gate.ac.uk>
- **Jappec** is an optimizing compiler that translates the grammar to **Java** source code:
  - The actual compiler is a standalone executable written in Haskell
  - **GATE** processing resource (PR) is provided that wraps the compiler and translates the grammar under the hood
- **Jappec** accepts the same syntax as the original **JAPE** implementation provided in **GATE 3.1+**
- Jappec can be used to write tokenisers as well as transducers

# What Japec does?

---

- The JAPE grammar is a set of rules
  - Each rule has a regular expression pattern on its left hand side
  - The set of all patterns defines finite-state transducer
- The compiler (Japec) **builds the transducer graph**
- The standard algorithms for **determination and minimization** are used to optimize the produced transducer
- The **optimized transducer** is used to **produce an algorithm** for pattern matching for the concrete grammar
  - The algorithm is expressed in the **Japec language**
  - It is an intermediate domain-specific language for pattern matching over annotations

## What Japec does? (II)

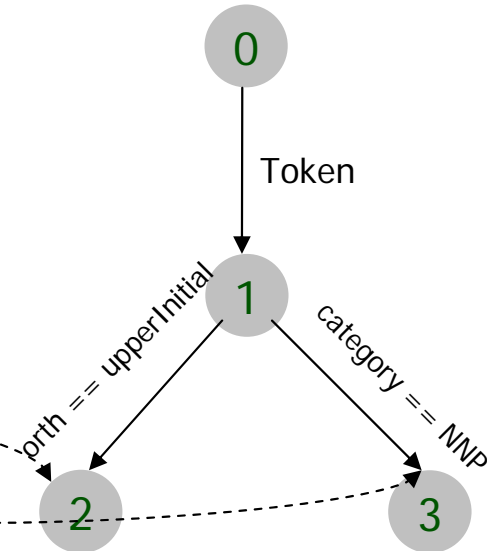
---

- From the intermediate language the concrete **Java source code** is produced:
  - The Java code is **compiled run-time** and can be used from the JAPEC transducer (which is a **GATE PR**)
- Generators for other languages are also possible.

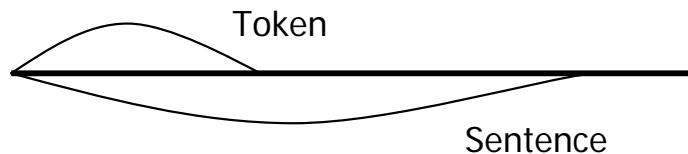
# Specifics in the Annotation Matching

- The input alphabet is set of **annotations** instead of symbols, so the produced transducer is always **nondeterministic** even after the classic determination algorithm
  - A given annotation can follow more than one route in the graph

Token	
orth	upperInitial
category	NNP



- At given position in the text more than one annotation can exist



# Extensions to JAPE syntax

---

- **Japec** allows pattern matching over the document text as well as over the annotation set. This makes it suitable to write tokenisers.

```
Rule: ShortNot1
  ("do" | "does"):verb ("n't"):not
  --> :verb.Token={kind=word, orth=lowercase}, :not.Token={kind=word,
  orth=apostrophe}
```

```
Rule: UpperInitial
  UPPERCASE_LETTER (LOWERCASE_LETTER
  (LOWERCASE_LETTER|DASH_PUNCTUATION|FORMAT)*)*
  --> :Token={orth=upperInitial, kind=word}
```

- Rule that covers the entire text matched in the left hand side.

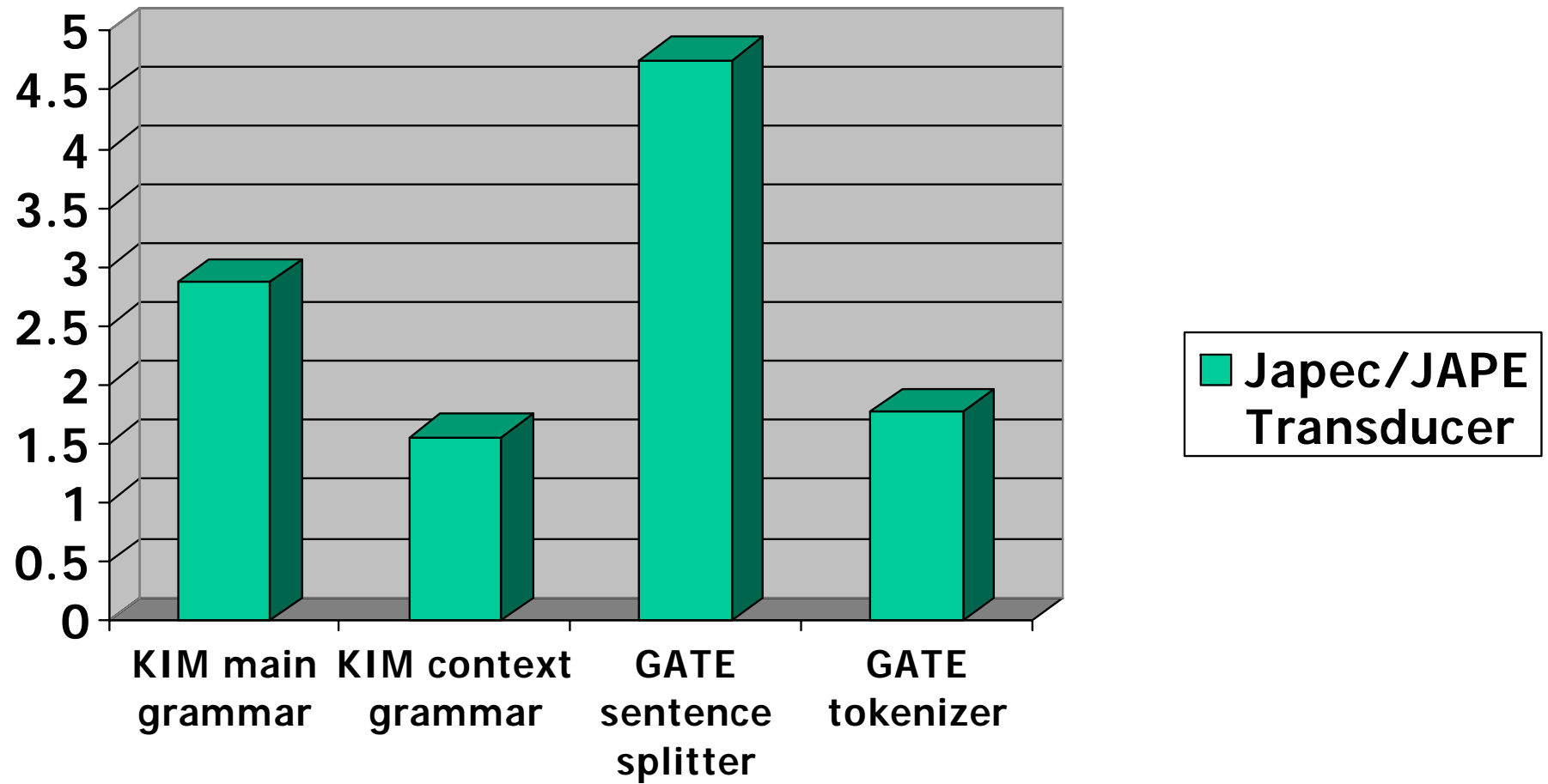
```
Rule: test
  ({Token} {Token})
  --> :TwoTokens={rule=test}
```

instead of:

```
Rule: test
  ({Token} {Token}):all
  --> :all.TwoTokens={rule=test}
```

# Performance

---



The relative performance of Japec vs. the usual JAPE transducer

# Download

---

- <http://www.ontotext.com/gate/japec.html>
- Two distributions are available:
  - **Source code:** japec-v0.2-src.zip
  - **Binary distribution:** japec-v0.2.zip
  - **Readme** file included in both distributions
- Licencing: **LGPL**
- Future plans:
  - The code of Japec is donated to the GATE team. They have adopted Japec internally, and are now working with Ontotext to port Japec to Java for release 4 of the system.
  - There are plans for re-implementation of the current version in Java and combination/consolidation with other related efforts

# Give Japec a Try!

---

For our industrial applications and products based on GATE, we needed faster JAPE transducers. We implemented **Japec**, because

**Japec speeds up any JAPE-heavy applications 2-5 times!**

<http://www.ontotext.com/gate/japec.html>