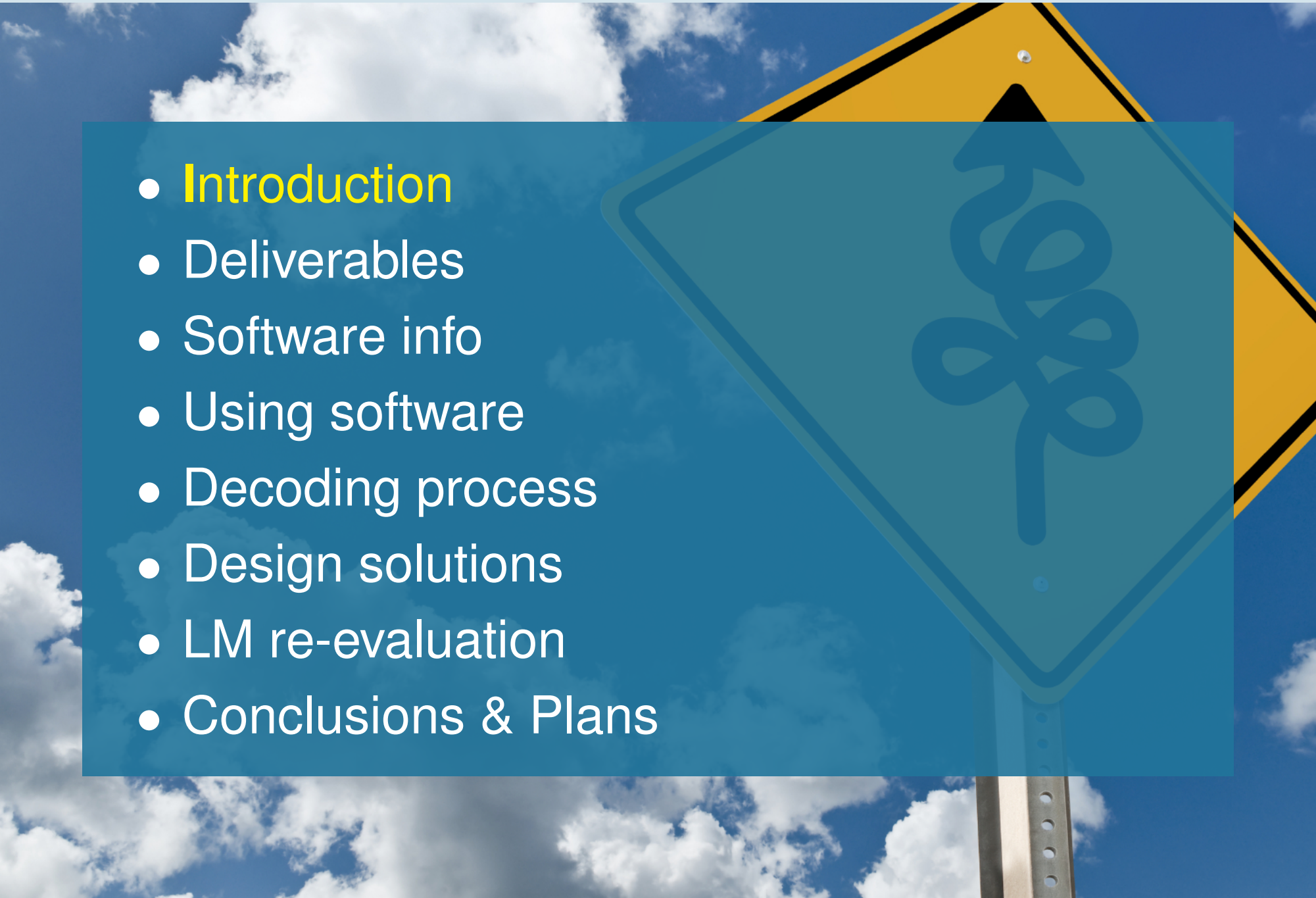
REMEDI: Robust and Efficient Machine Translation in a Distributed Infrastructure

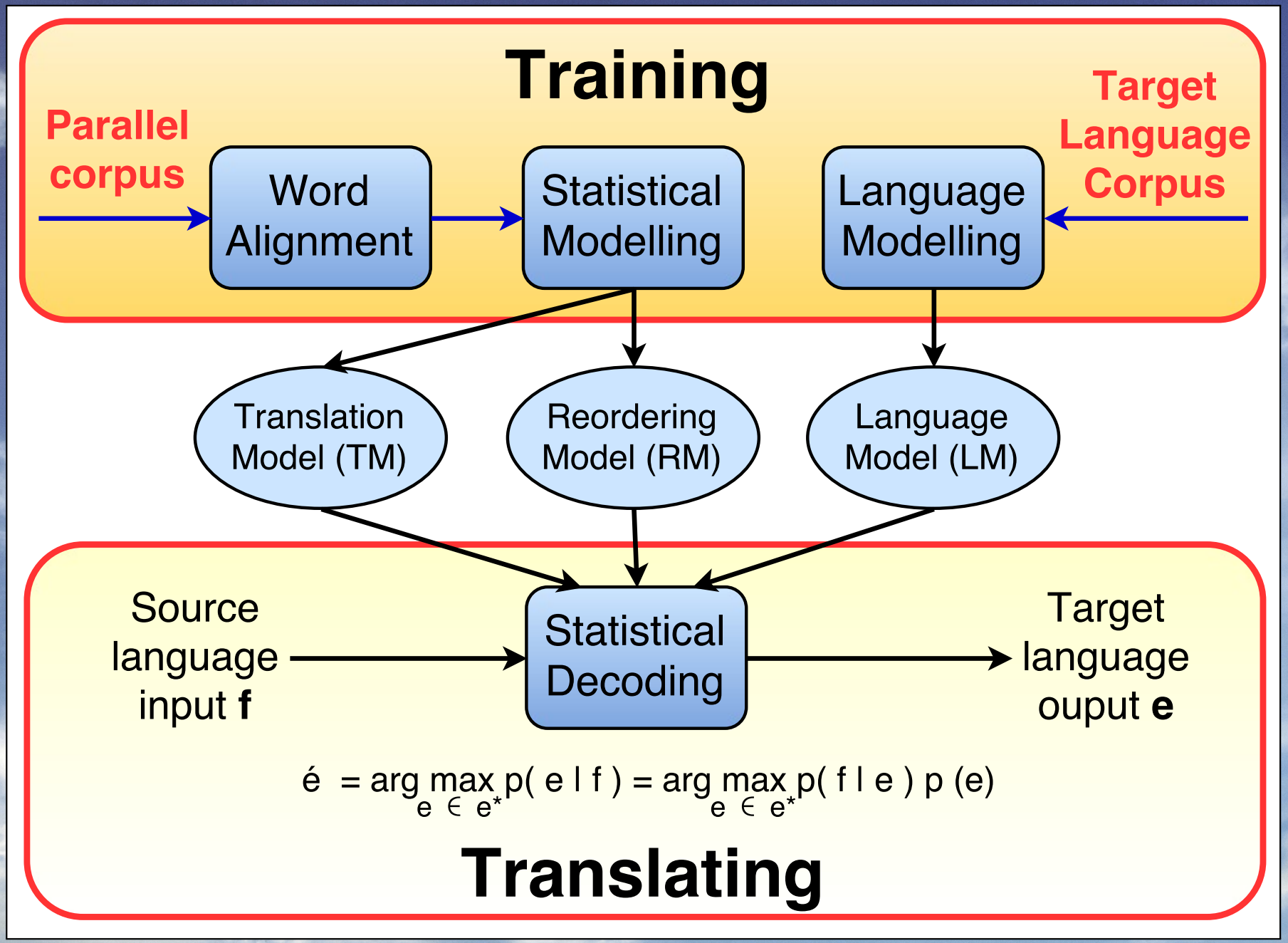
Bi-Annual Report: Month 12

Dr. Ivan S. Zapreev • UvA • ivan.zapreev@uva.nl
Dr. Christof Monz • UvA • c.monz@uva.nl

Table of Contents



Introduction: Statistical Machine Translation



Introduction: Expected Month 12 delivery

- S1, WP1:
  1. Translation model
  2. Reordering model
  3. Decoder search
  4. Pruning strategies
- R2:
  - Month 12 report

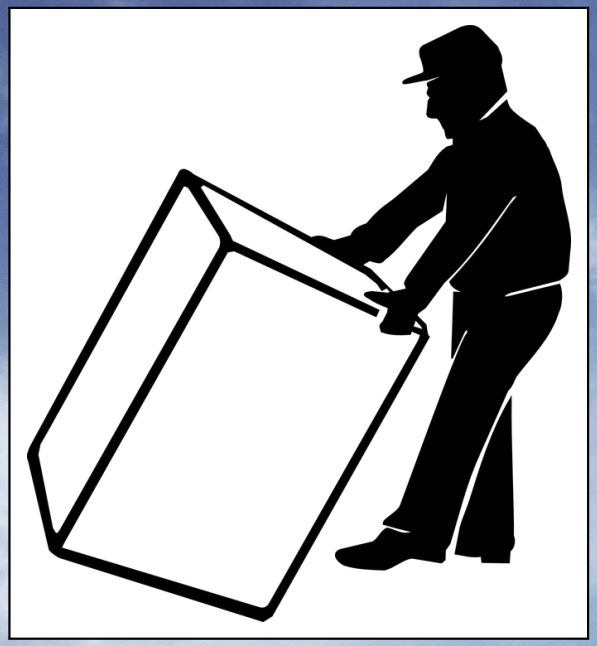
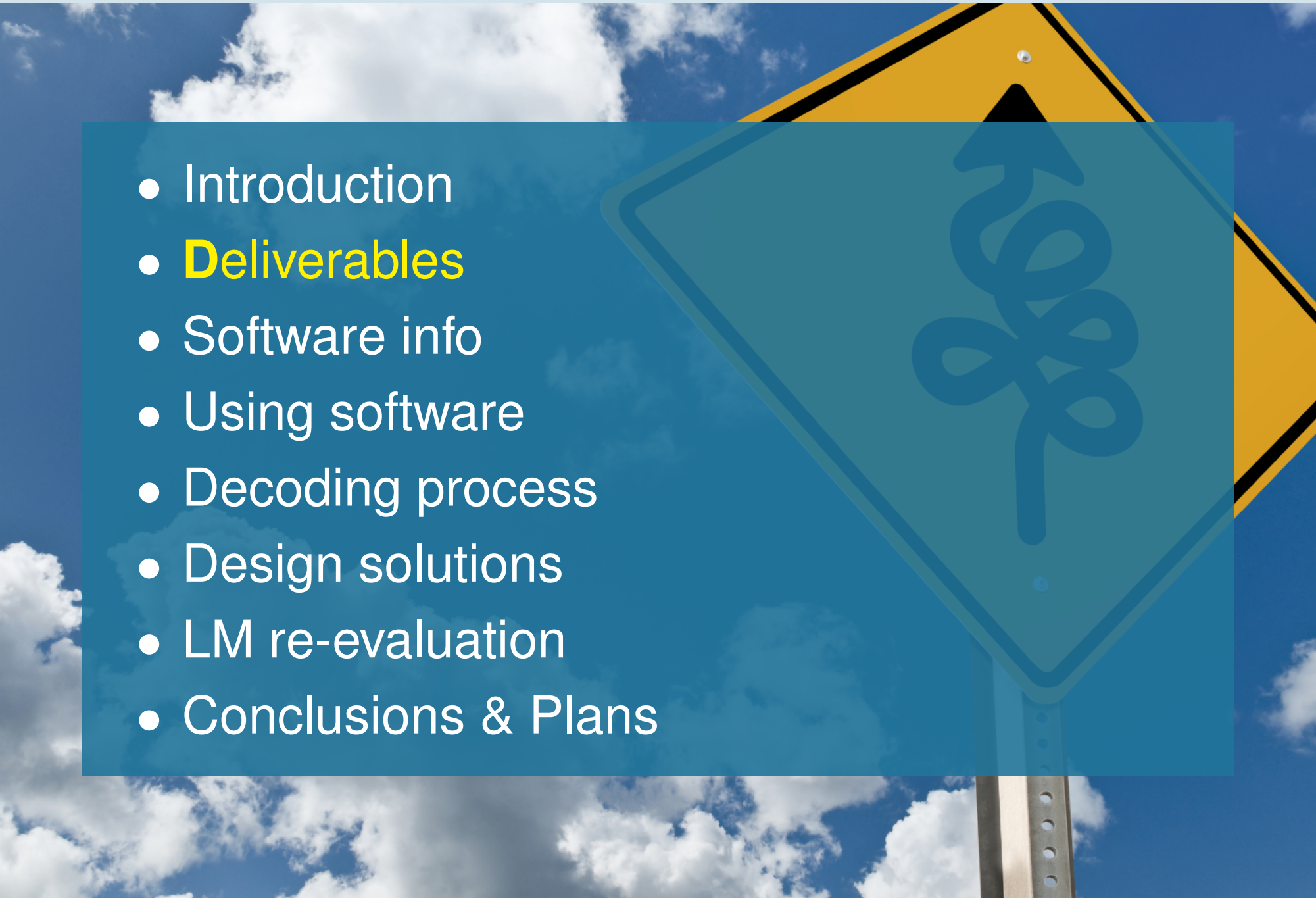


Table of Contents



Deliverable: Software, Report, Example models

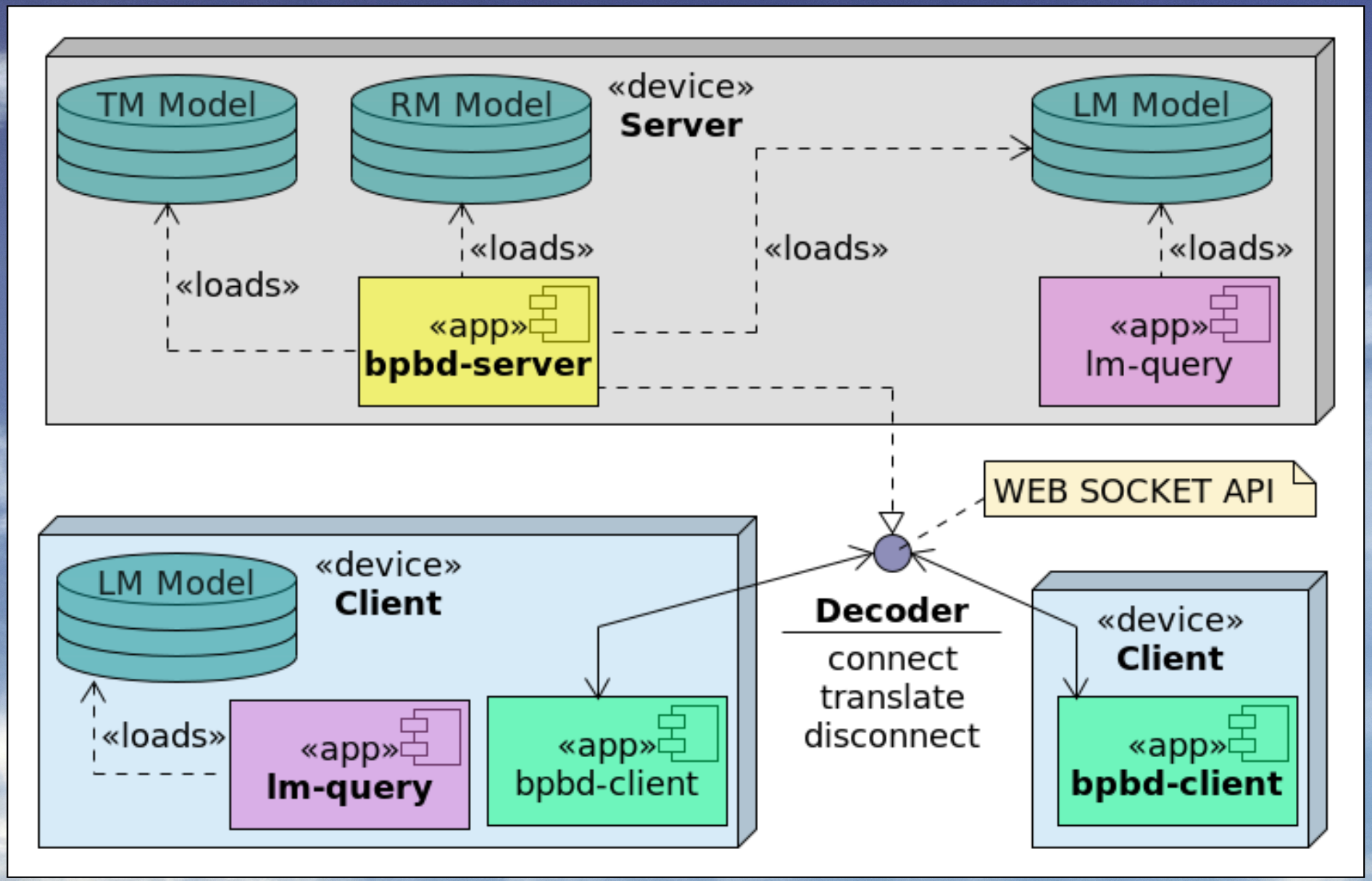
REMEDI/month-12/

- **software/**
  - doc/
  - ext/
  - inc/
  - src/
  - nbproject/
- **report/**
- **data/**

GitHub project:

`https://github.com/ivan-zapreev/Basic-Phrase-Based-Decoding`

Deliverable: Product & Executables



Deliverable: Software Functionality

MUSTs:

- Models:
  - Reordering
  - Translation
- Decoding:
  - Beam search
  - Future costs
  - Threshold
  - Histogram
  - Recombination

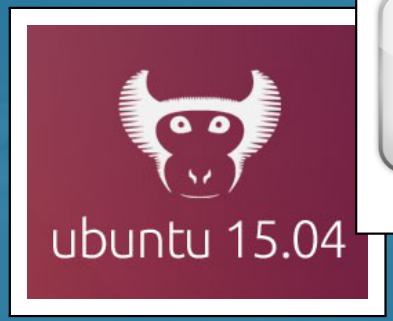
EXTRAs:

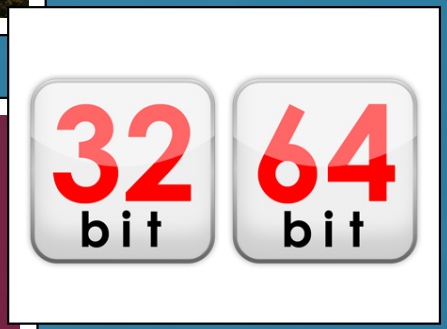
- Valgrind
- TM filtering
- RM filtering
- Multi-Threaded
- Distr. Design
- Server console
- Documentation

Table of Contents

Software info: Type, License, Platforms, Architectures

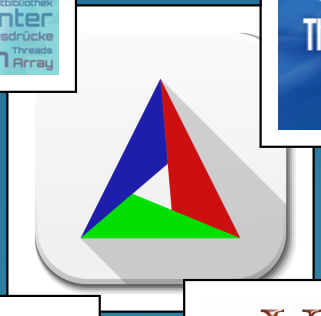
- Type:
  - Command line
- License:
  - GPL v2.0
- Platforms:
  - CentOS 6.6
  - Ubuntu 15.04
  - Mac OS 10.10
- Architectures:
  - 64-bit - full
  - 32-bit - partial





Software info: Language, Project, Docs, Testing

- Language:
  - C++ 11
- Project:
  - Netbeans 8.0.2
  - cmake
- Documentation:
  - Doxygen
- Profiling:
  - Valgrind



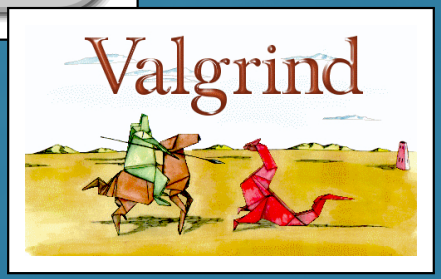
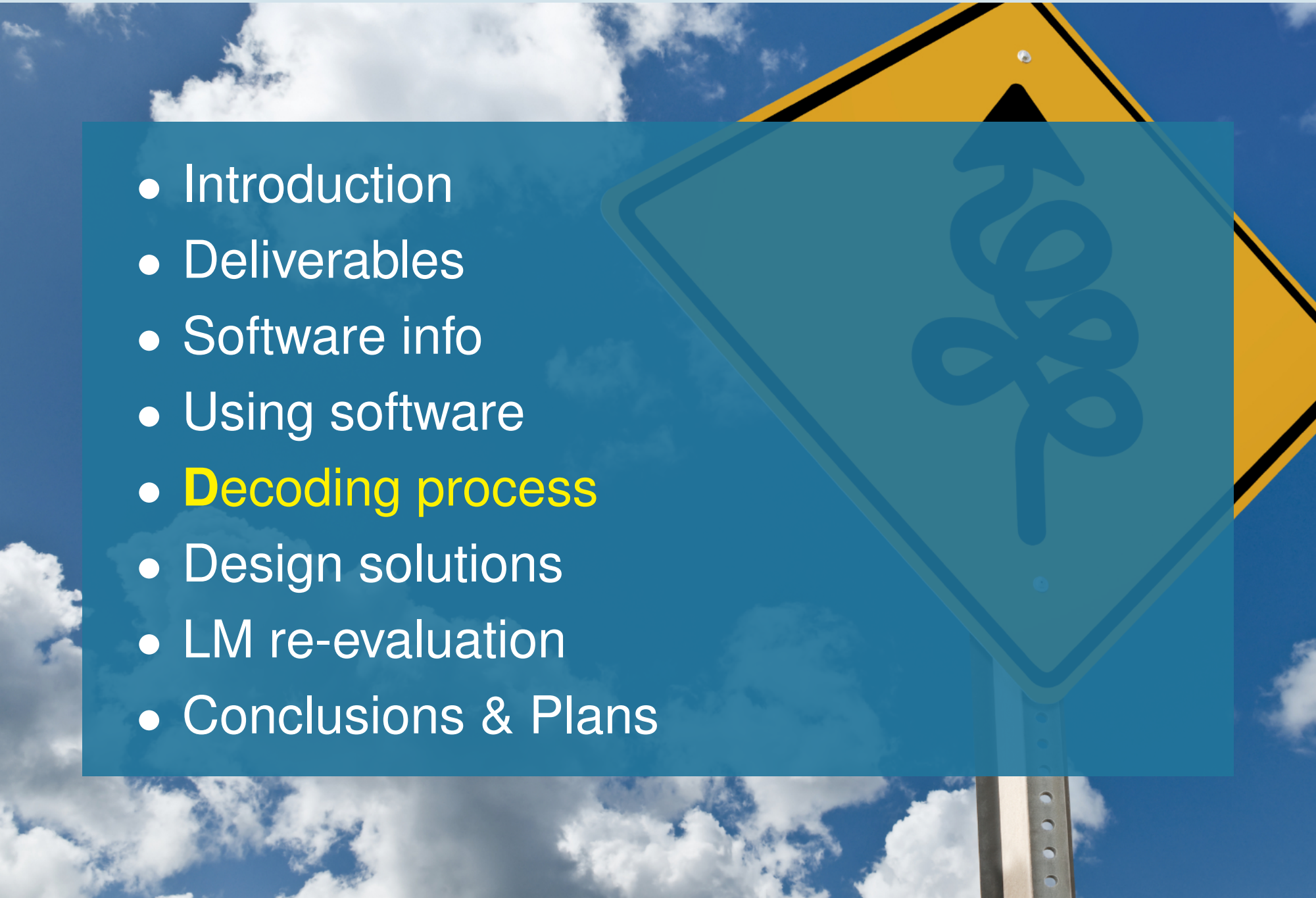


Table of Contents

Using software: Building & Running

Table of Contents



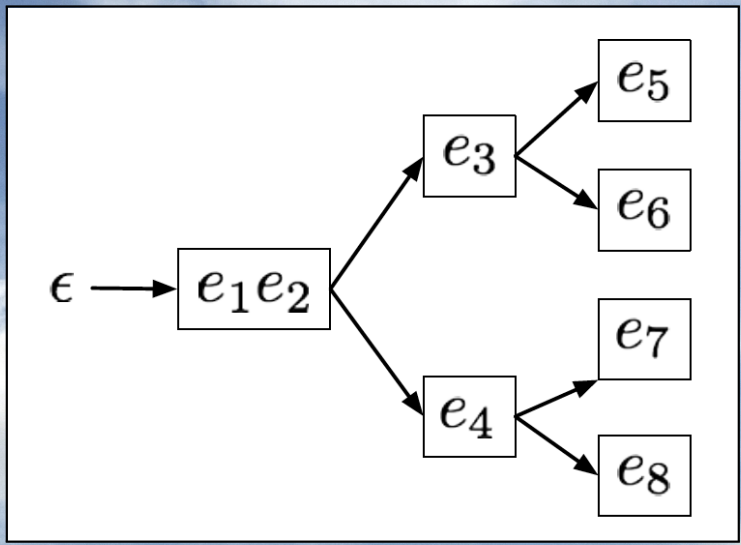
Decoding process: Search Space

$$\text{trans}(f) = \arg\max_{e \in \tilde{E}} p(f|e) \cdot p(e)$$

Problems:

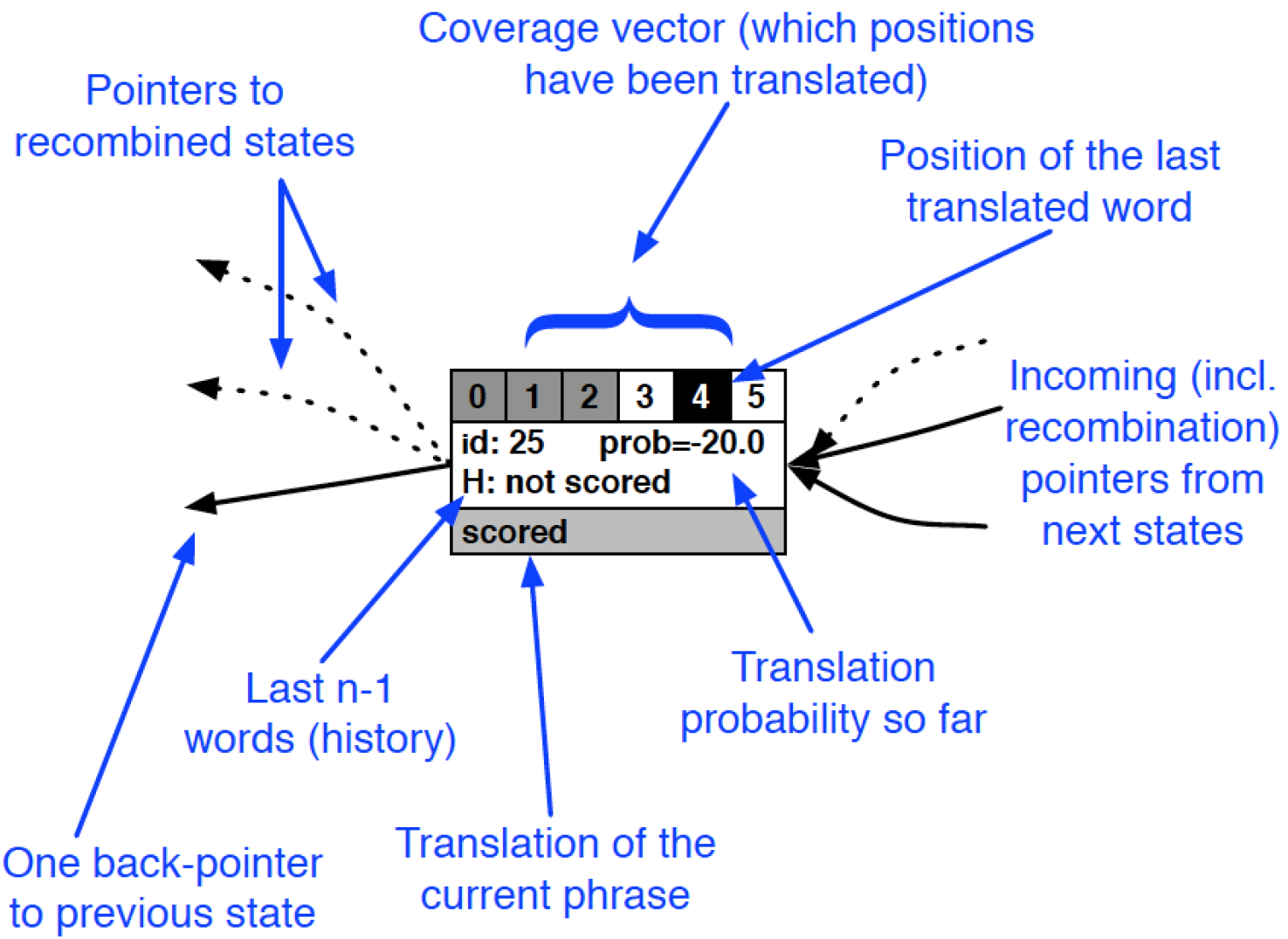
- Huge state space, $E^* = \Sigma_{i=0}^{\infty} V^i$
- Huge time complexity, $O(n!)$

$$\tilde{E} = \left\{ \begin{array}{l} e_1 e_2 e_3 e_5, \\ e_1 e_2 e_3 e_6, \\ e_1 e_2 e_4 e_7, \\ e_1 e_2 e_4 e_8 \end{array} \right\}$$

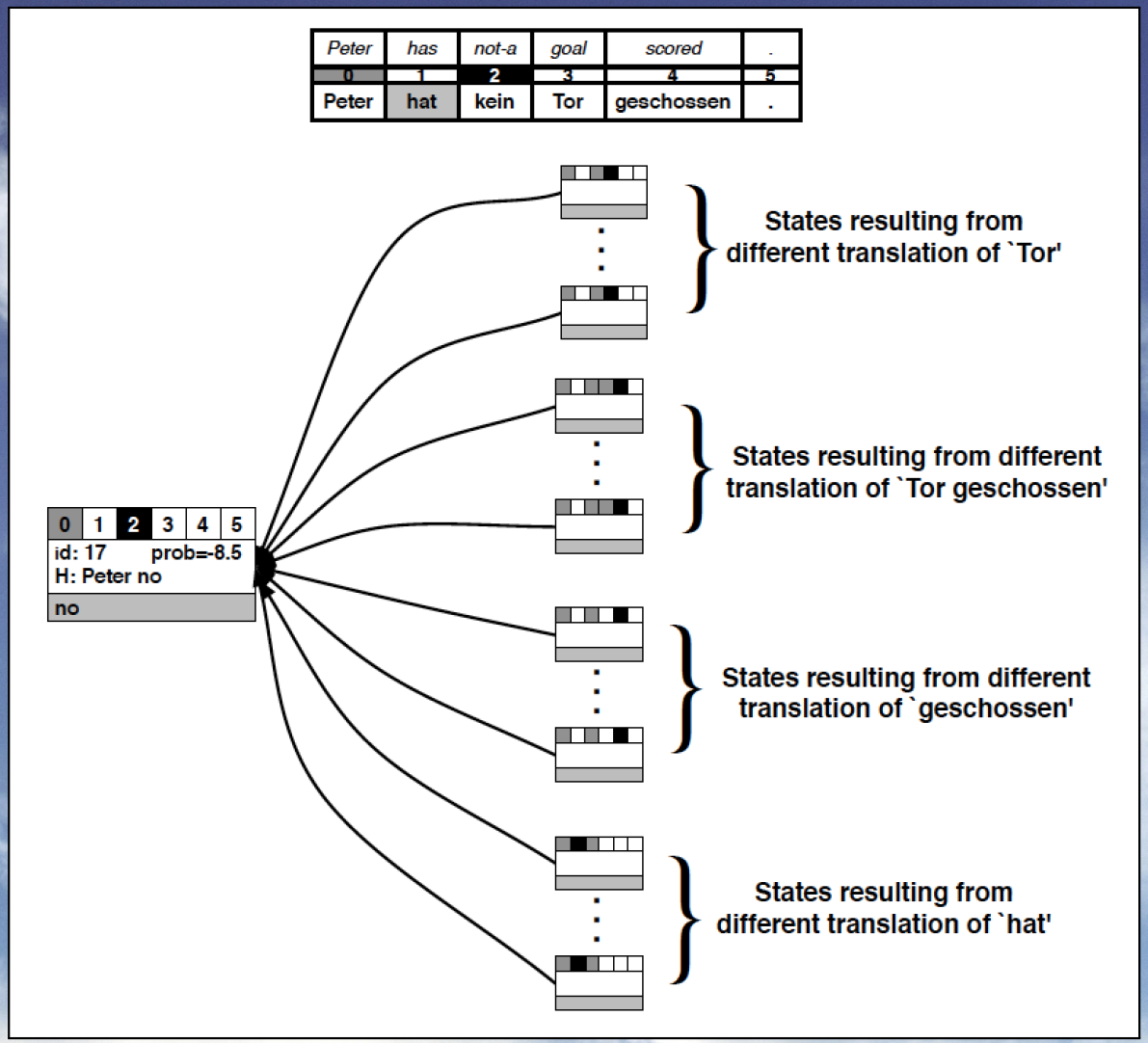


Decoding process: Decoding State

Peter	*has*	*not-a*	*goal*	*scored*	.
0	1	2	3	4	5
Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**



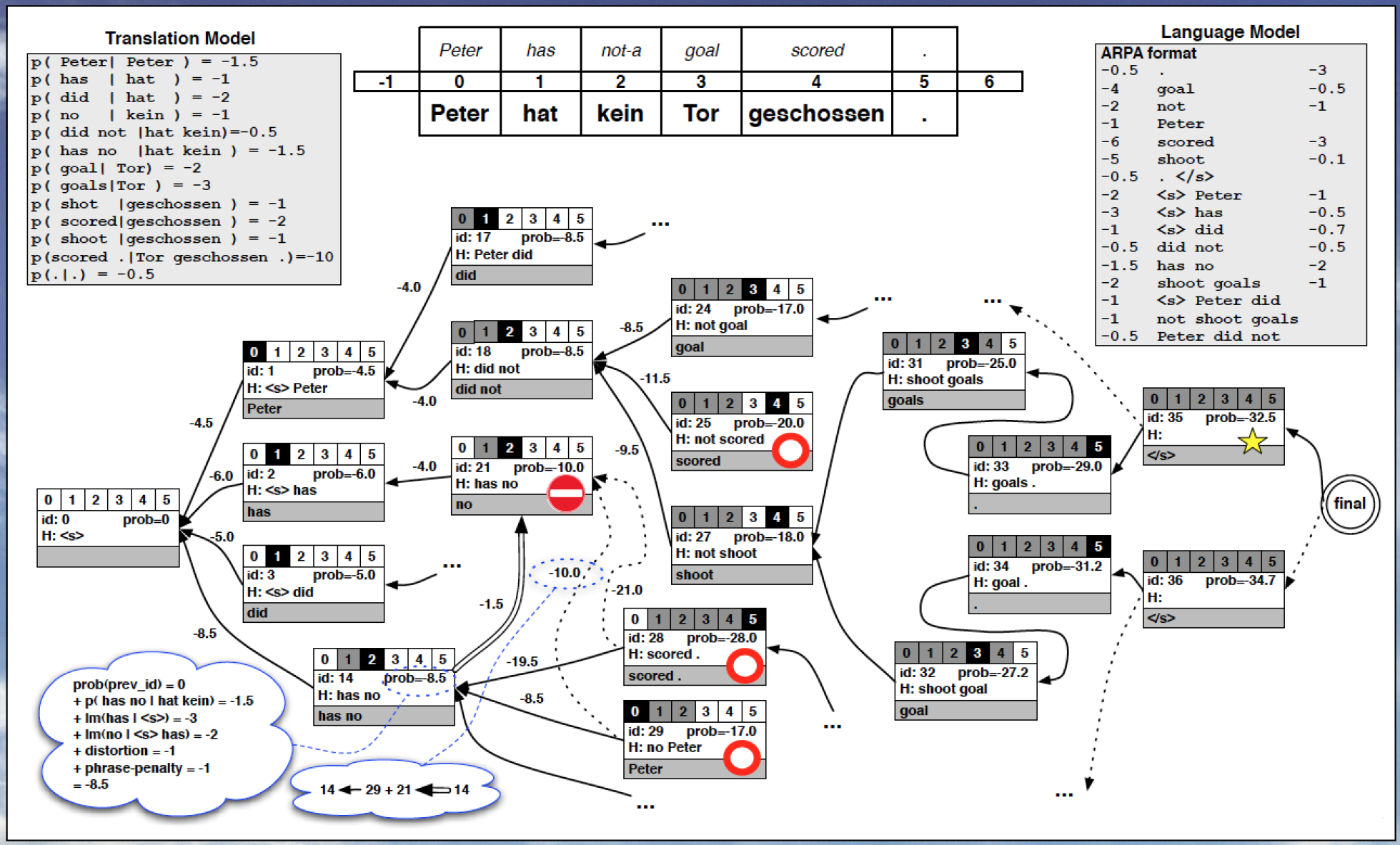
Decoding process: State Expansion



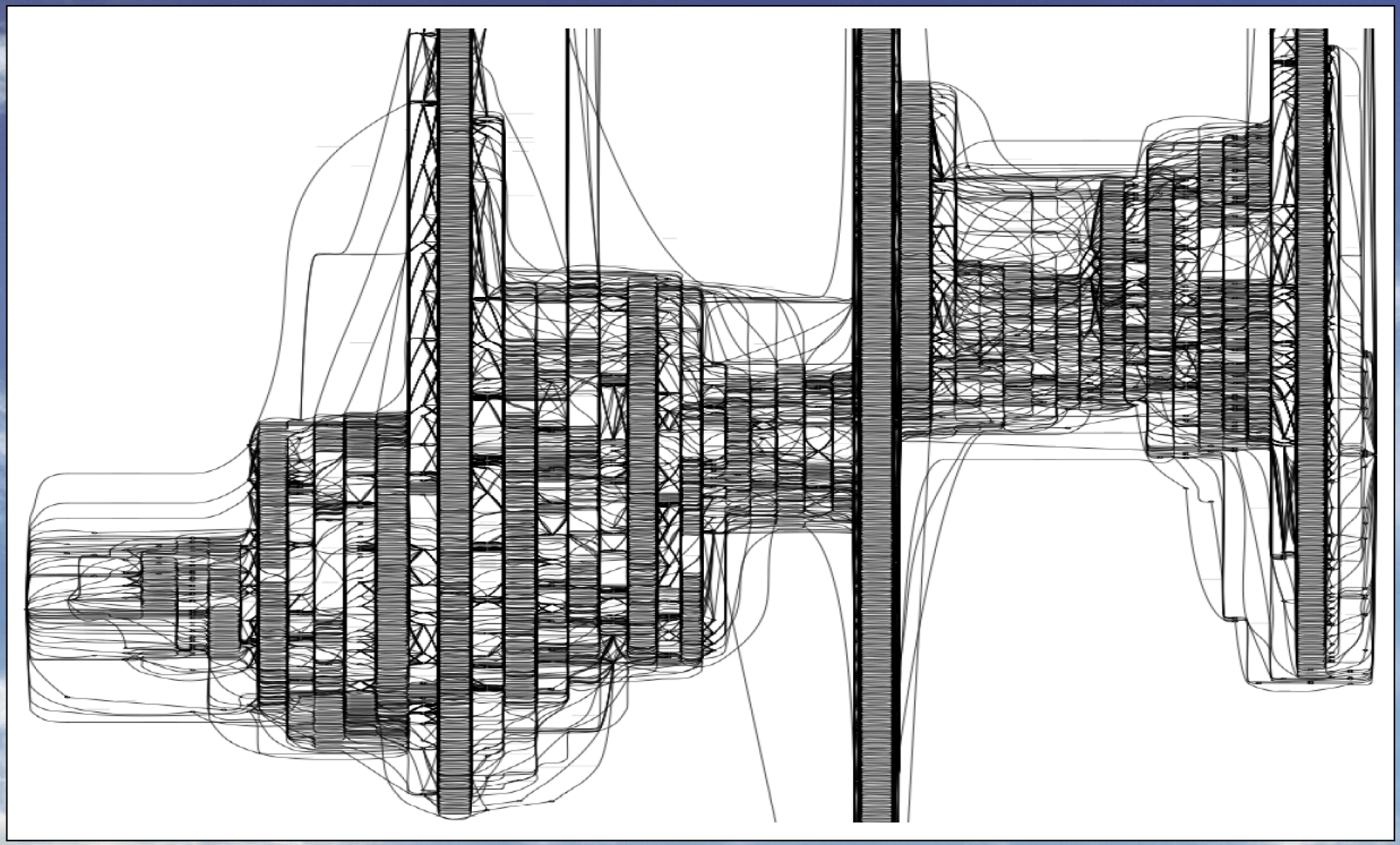
Limit:

Distortion; # Translations; Min translation probability

Decoding process: Decoding Example

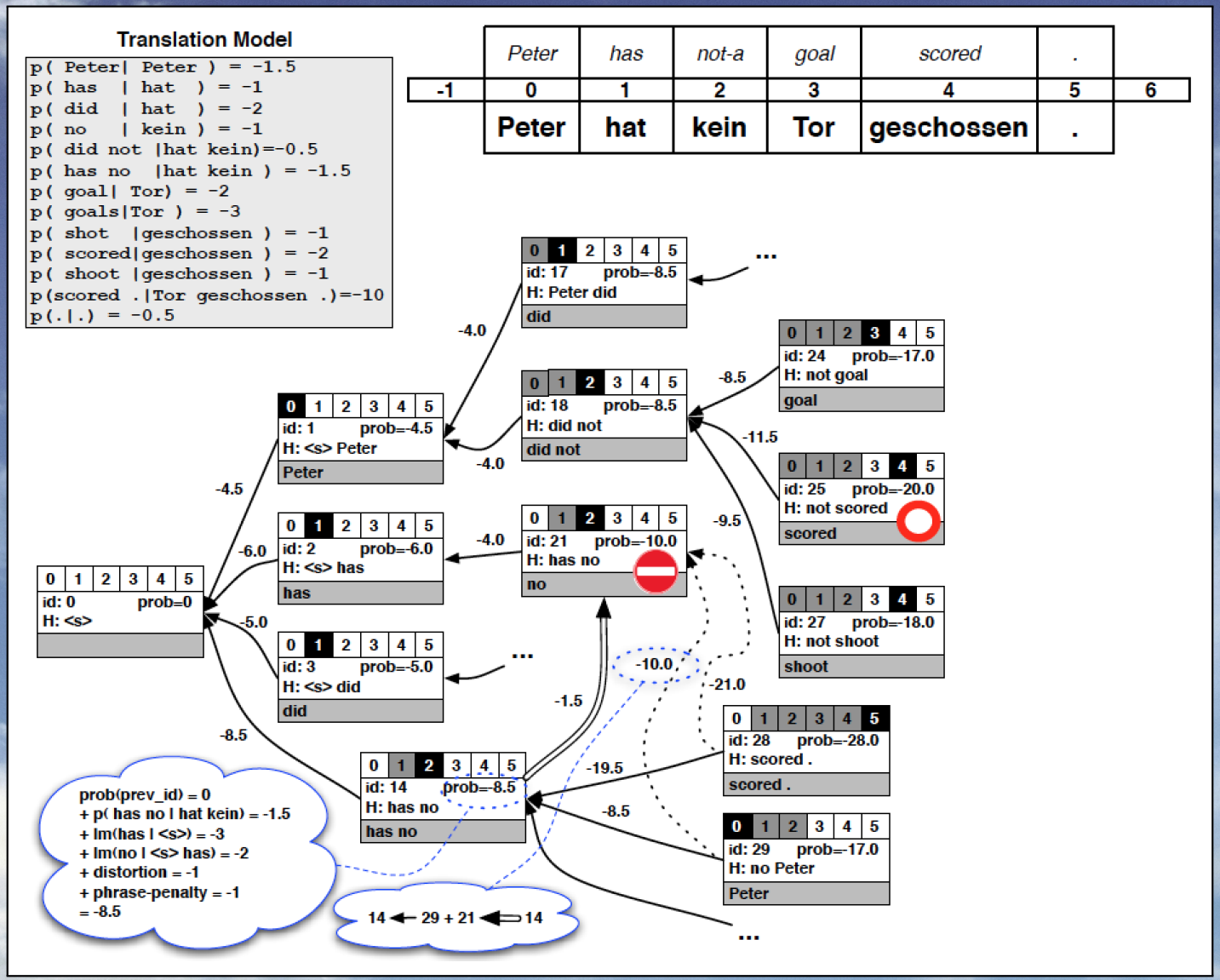


Decoding process: The Real Thing



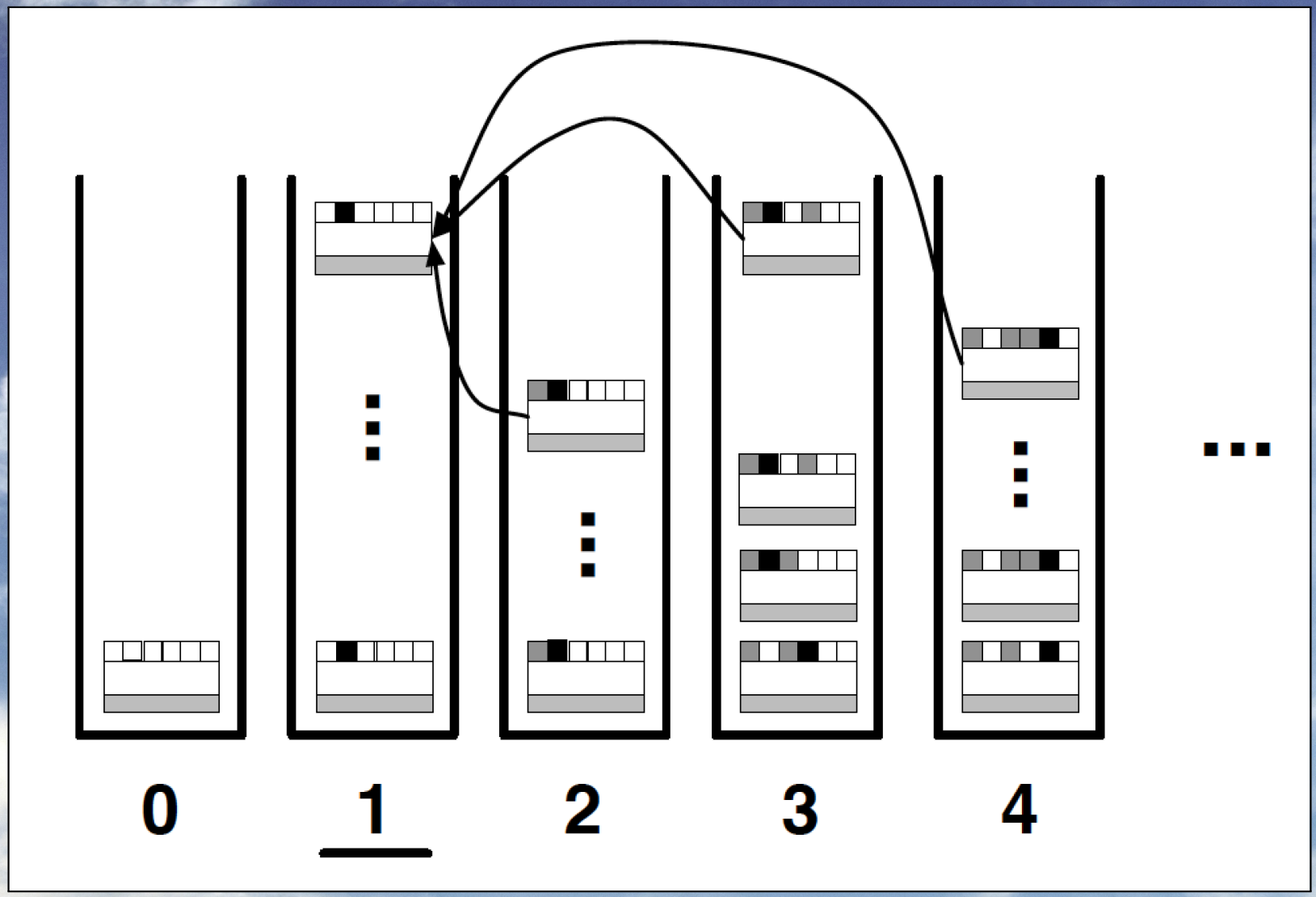
Decoding process: Recombination

Recombine states ensuring the same future expansion tree;



Decoding process: Multi-Stack Decoding

Comparable hypothesis - the same stack; Incremental;



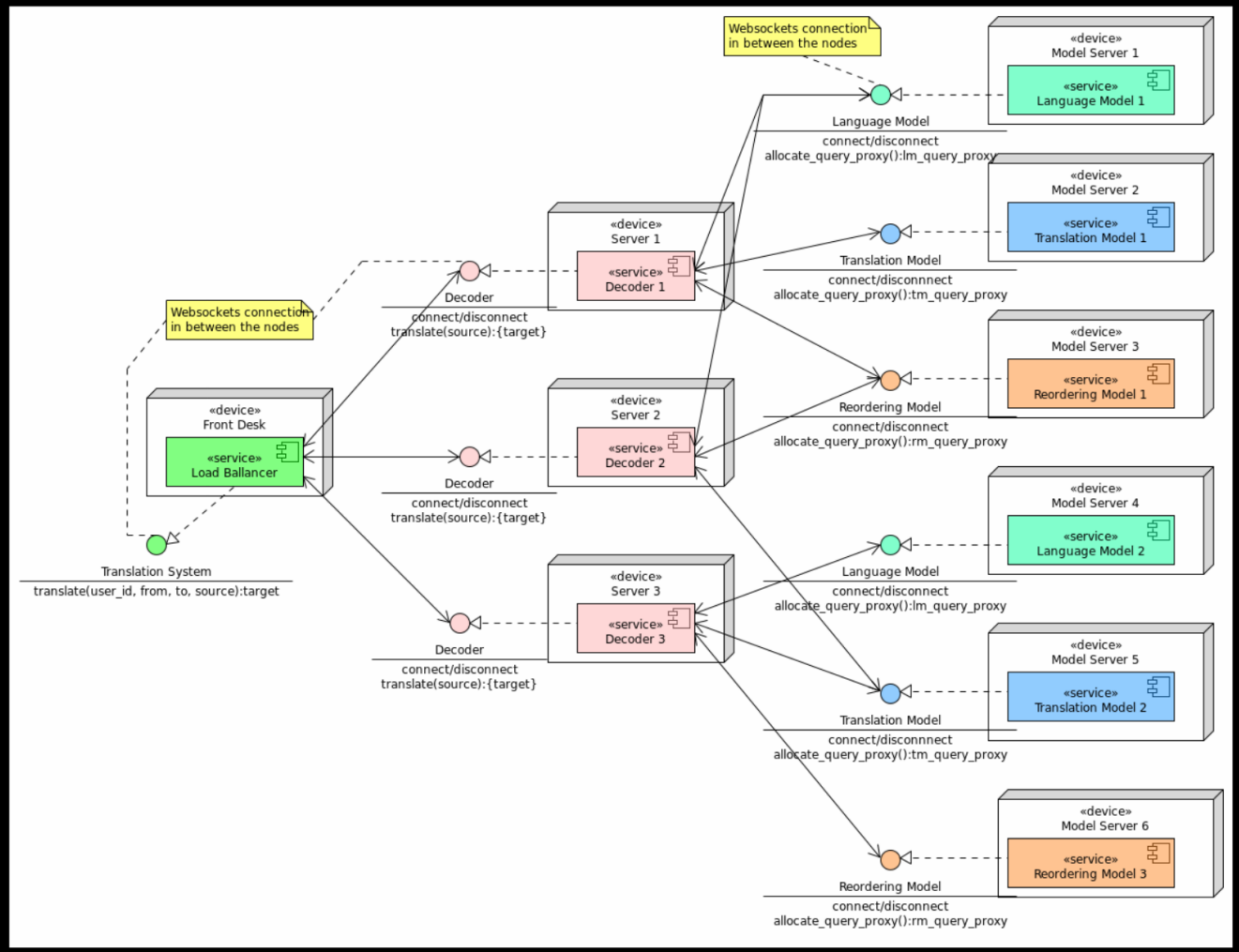
Decoding process: Beam Search Pruning

- **Histogram pruning:**
  - Limit the number of states per stack
- **Threshold pruning:**
  - Threshold $0 < \theta < 1$ times best cost
    - Current log costs:
      $= p_{TM}(e|f) + p_{RM}(e|f) + p_{LM}(e) + wp + pp + dp$
    - Future log costs:
      $\approx max_{e \in \tilde{E}} (p_{TM}(e|f) + p_{LM}(e))$

Table of Contents

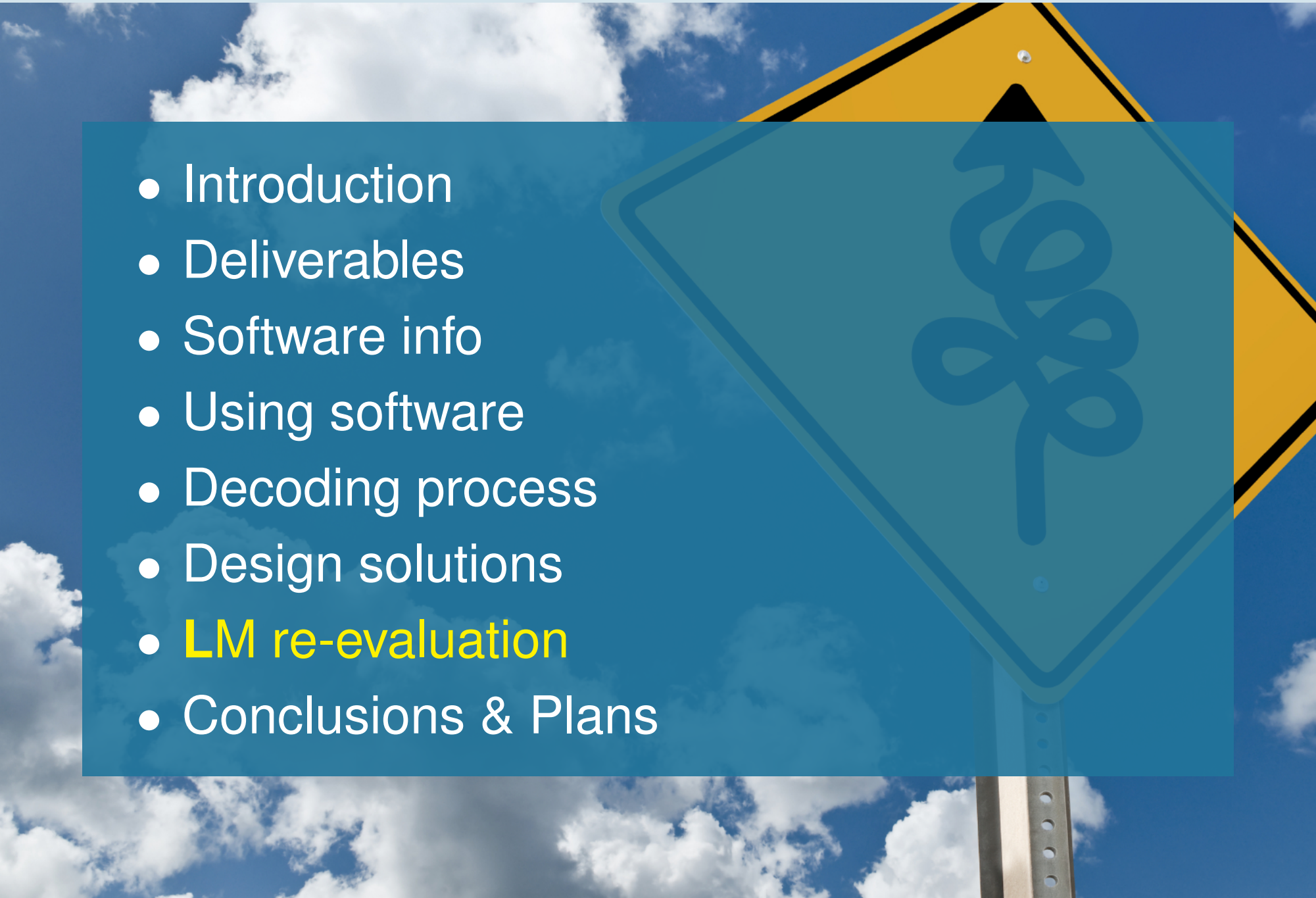
Design solutions: Prezi Presentation

The ultimate design



First implementation

Table of Contents



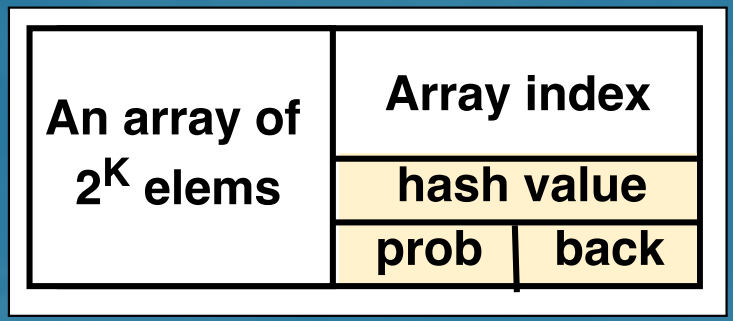
Algorithms: Additional LM improvements

General:

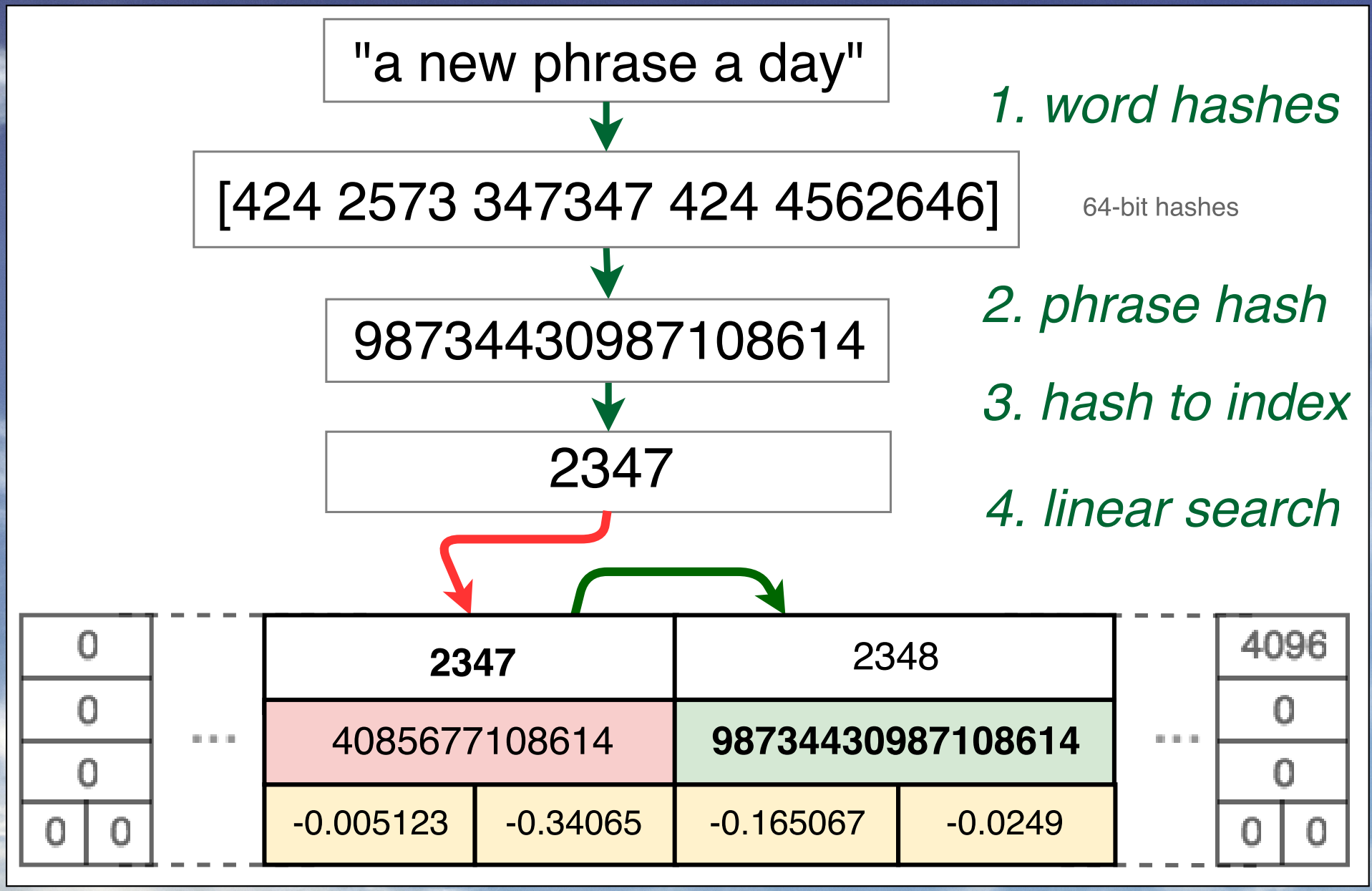
- LM multi-threading
- Joint LM probs
- Optimizations
- Re-evaluation

A KenLM-like trie:

- Linear Probing Hash Map
- Time complexity:
  - Max: $o(N)$
  - Min: $o(1)$
  - Avg: $o(1)$
- $K = log_2(\#Factor * (N + 1))$
- $\#Factor \approx 2$



Algorithms: A new LM model trie (h2dm)

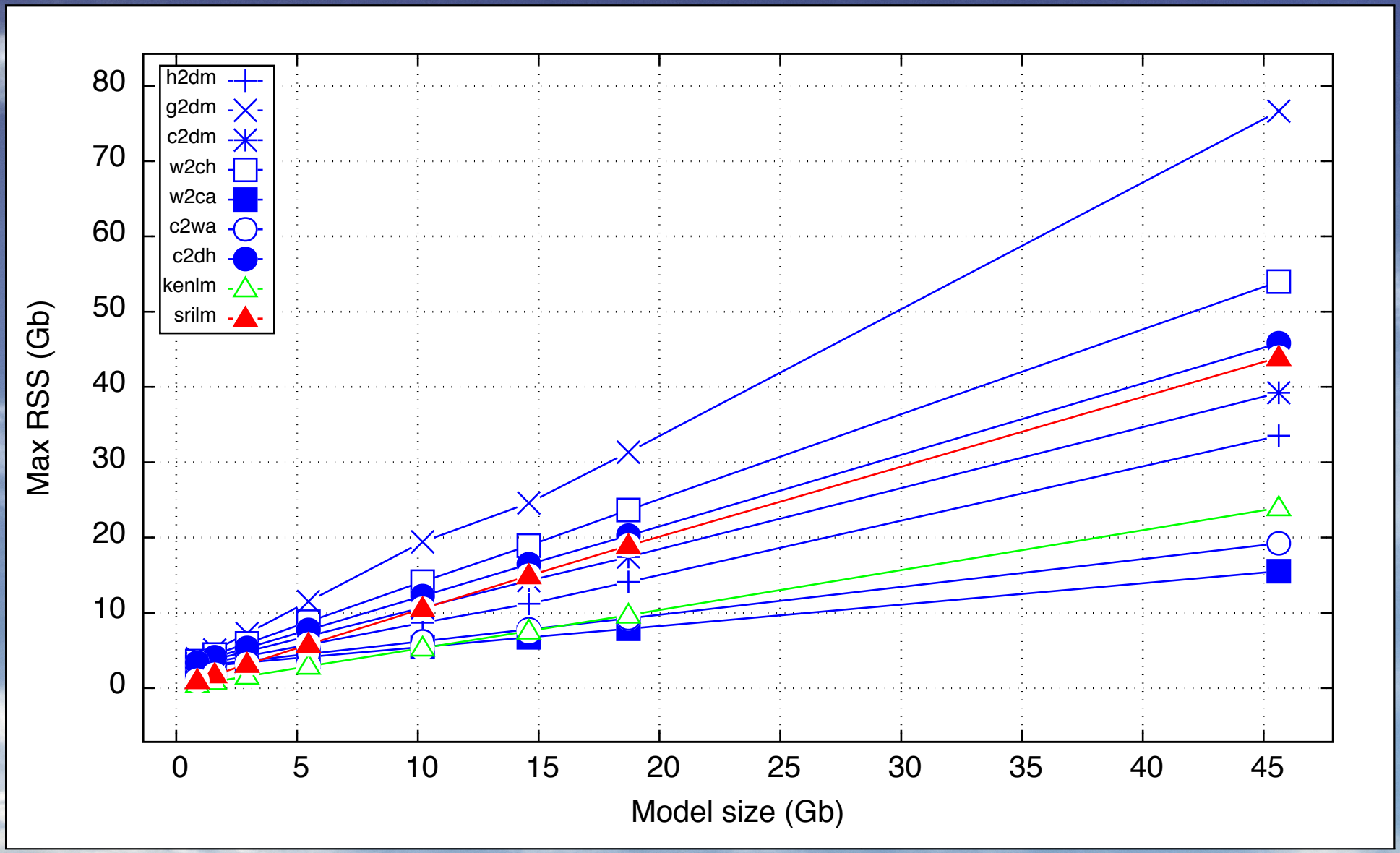


Considered tools and configurations.

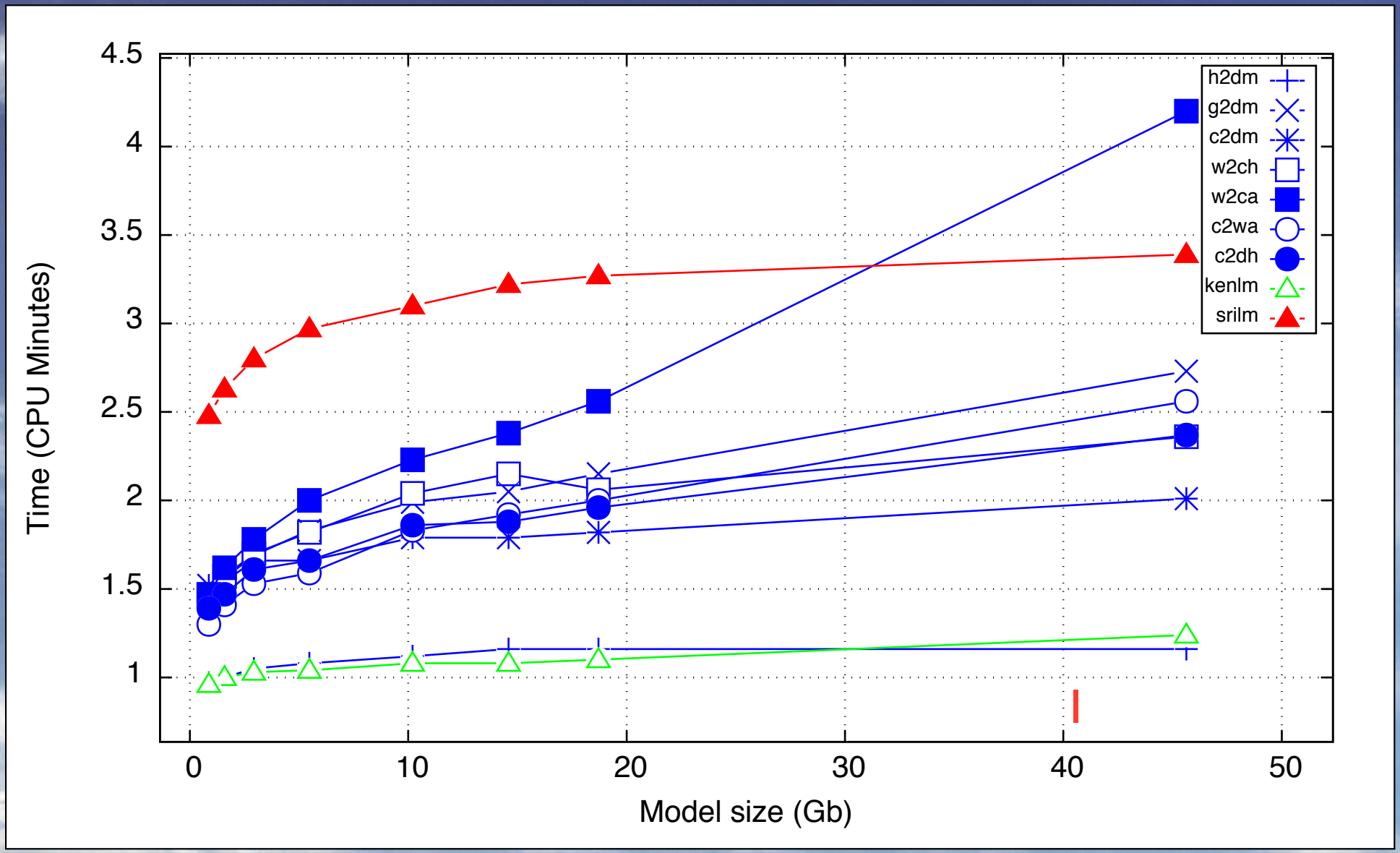
- Tools:
  - Our tool:
    - lm-query release, v1.0
    - Different trie types
  - KenLM:
    - GitHub snapshot
    - Modified source code
  - SRILM:
    - v1.7.0
- Target:
  - Model MRSS
  - Query CPU times
- Config:
  - Various 5-gram model sizes
  - 10^8 of 5-gram queries

100% fair!

MRSS: Maximum Resident Memory Size



CPU: CPU execution times



Experiments: Summary

Table: MRSS	
65%	W2CA
80%	C2WA
100%	**KenLM**
140%	**H2DM**
164%	C2DM
183%	**SRILM**
191%	C2DH
225%	W2CH
320%	G2DM

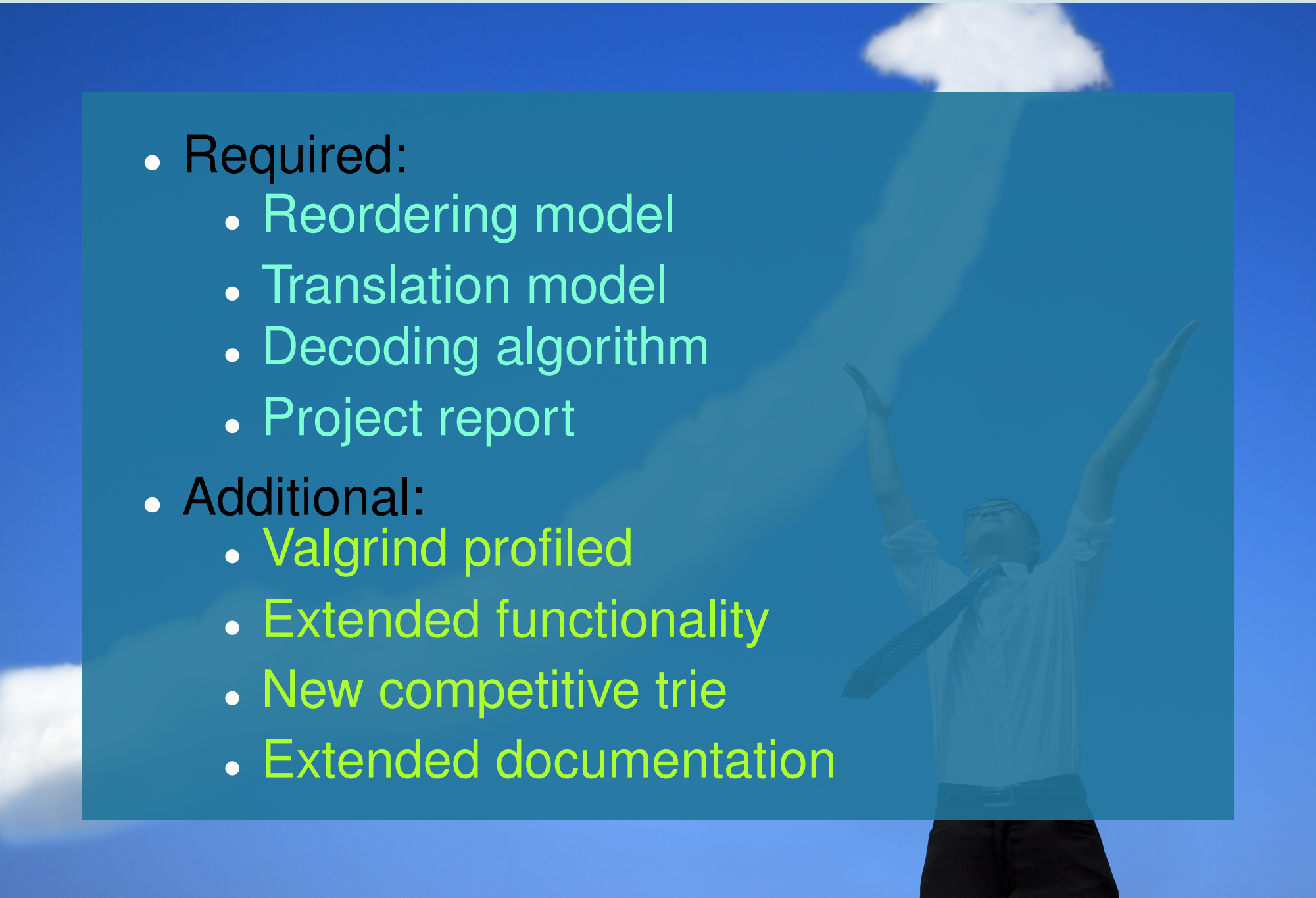
Table: CPU	
94%	**H2DM**
100%	**KenLM**
162%	C2DM
190%	W2CH
191%	C2DH
206%	C2WA
220%	G2DM
273%	**SRILM**
339%	W2CA

Table: TOTAL	
100%	**KenLM**
117%	**H2DM**
143%	C2WA
163%	C2DM
191%	C2DH
202%	W2CA
208%	W2DH
228%	**SRILM**
270%	G2DM

Largest model || KenLM - 100% || $TOTAL = \frac{(MRSS + CPU)}{2.0}$

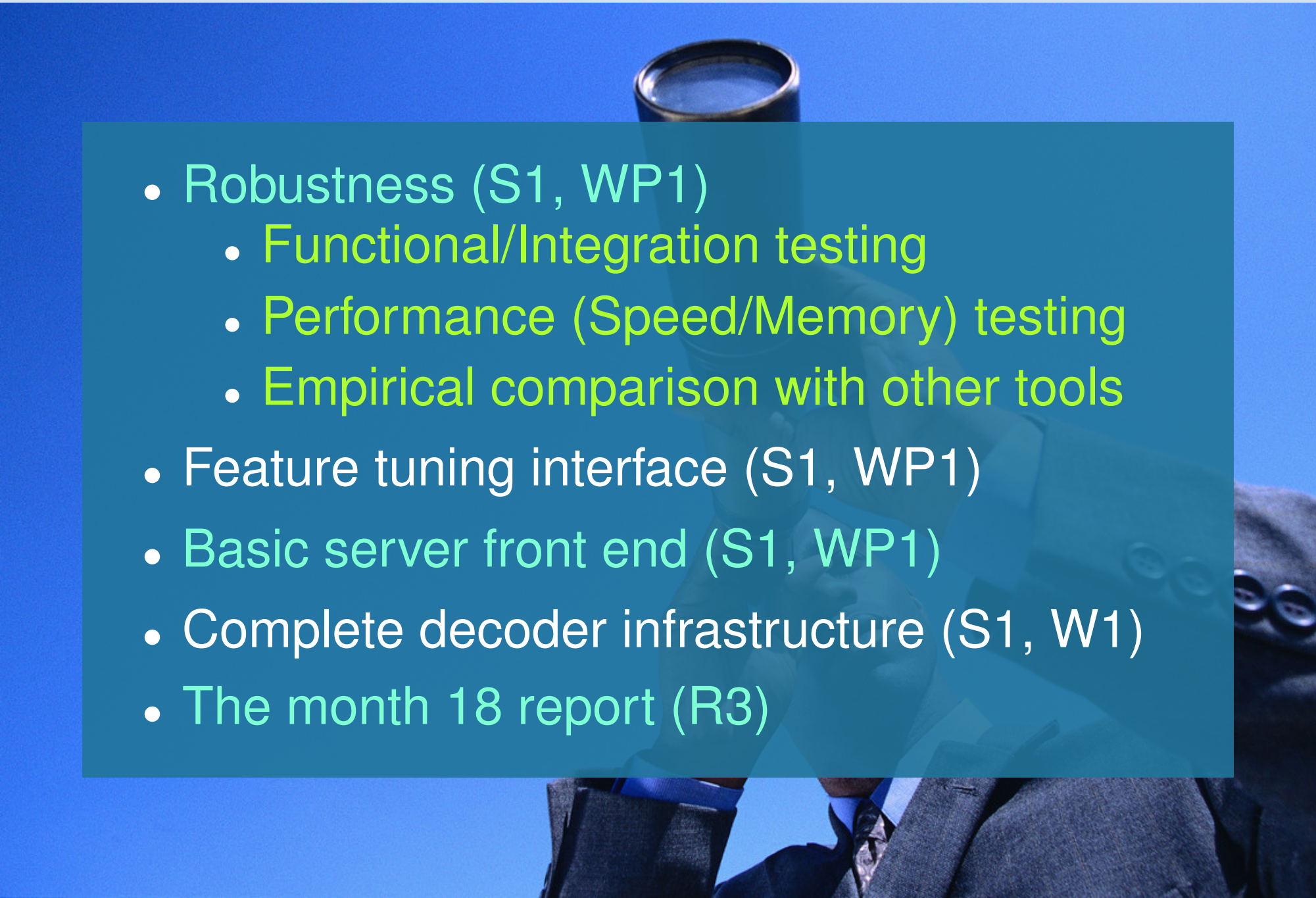
Table of Contents

Conclusions



- Required:
  - Reordering model
  - Translation model
  - Decoding algorithm
  - Project report
- Additional:
  - Valgrind profiled
  - Extended functionality
  - New competitive trie
  - Extended documentation

Future plans



- Robustness (S1, WP1)
  - Functional/Integration testing
  - Performance (Speed/Memory) testing
  - Empirical comparison with other tools
- Feature tuning interface (S1, WP1)
- Basic server front end (S1, WP1)
- Complete decoder infrastructure (S1, W1)
- The month 18 report (R3)

Discussion

Thank you all!