Trellis: A Domain-Specific Language for Hidden Markov Models with Sparse Transitions

Lars Hummelgren, Viktor Palmkvist, Linnea Stjerna, Xuechun Xu, Joakim Jaldén, David Broman

Outline

- Introduction
- Hidden Markov Models
- Trellis
- Results
- Conclusion

- **Introduction**
- Hidden Markov Models
- Trellis
- Results
- Conclusion

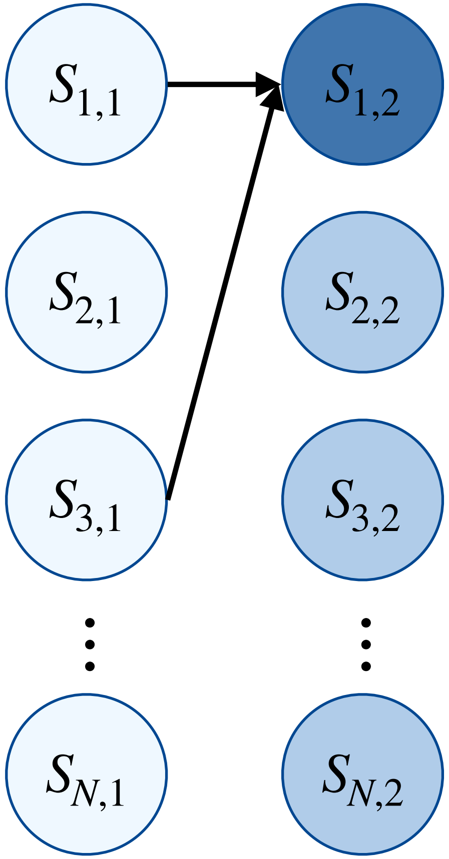
Introduction

Motivation

- Hidden Markov Models (HMMs) are used in many different fields
  - Bioinformatics
  - Signal processing
  - Pattern recognition

Problem

- Sparse HMMs with many states
- Model separation



Related Work

- **Problem**: Focused on dense HMMs

HMMoC (Lunter, 2007)

HMMConverter (Lam & Meyer, 2009)

zipHMM (Sand et. al., 2013)

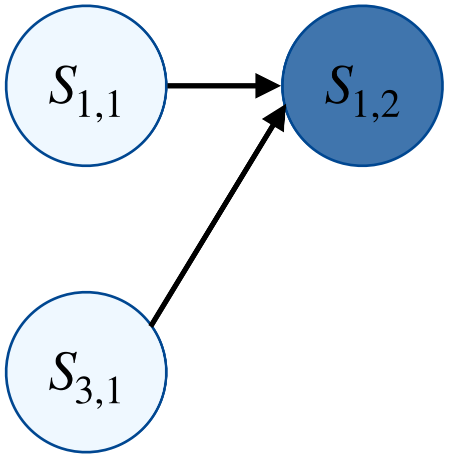
StochHMM (Lott & Korf, 2014)

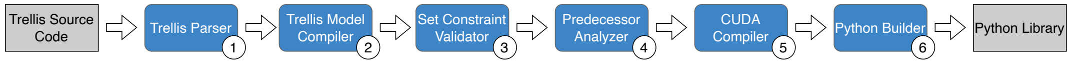
pomegranate (Schreiber, 2018)

Contributions

- Trellis DSL for sparse HMMs
- Trellis compiler producing efficient GPU code
- Predecessor computation

```
let merlength = 3
let maxduration = 16
type Nucleotide = {A, C, G, T}
alias Kmer = Nucleotide[merlength]
alias Duration = 1 .. maxduration
alias ObsType = 0 .. 100
model {
 state = (Duration, Kmer)
 output = ObsType
 table initialProb(Duration, Kmer)
 table outputProb(ObsType, Kmer)
 table trans1(Kmer, Nucleotide)
 table trans2(Duration)
 table gamma()
 P(initial x) = initialProb(x[0], x[1])
 P(output o | x) = outputProb(o, x[1])
 P(transition x y) = {
  | { (1, [a, as...]) -> (k, [bs..., b]) | as == bs } =>
   trans1(x[1], y[1][2]) * trans2(y[0])
  | { (n1, x1) -> (n2, x2) | x1 == x2, n1 == maxduration, n2 ==
maxduration } =>
   gamma()
  | { (n1, x1) -> (n2, x2) | x1 == x2, n1 == maxduration, n2 ==
maxduration - 1 } =>
   1.0 - gamma()
  | { (n1, x1) -> (n2, x2) | x1 == x2, n2 == n1 - 1, n2 !=
maxduration - 1 } =>
   1.0
 }
}
```





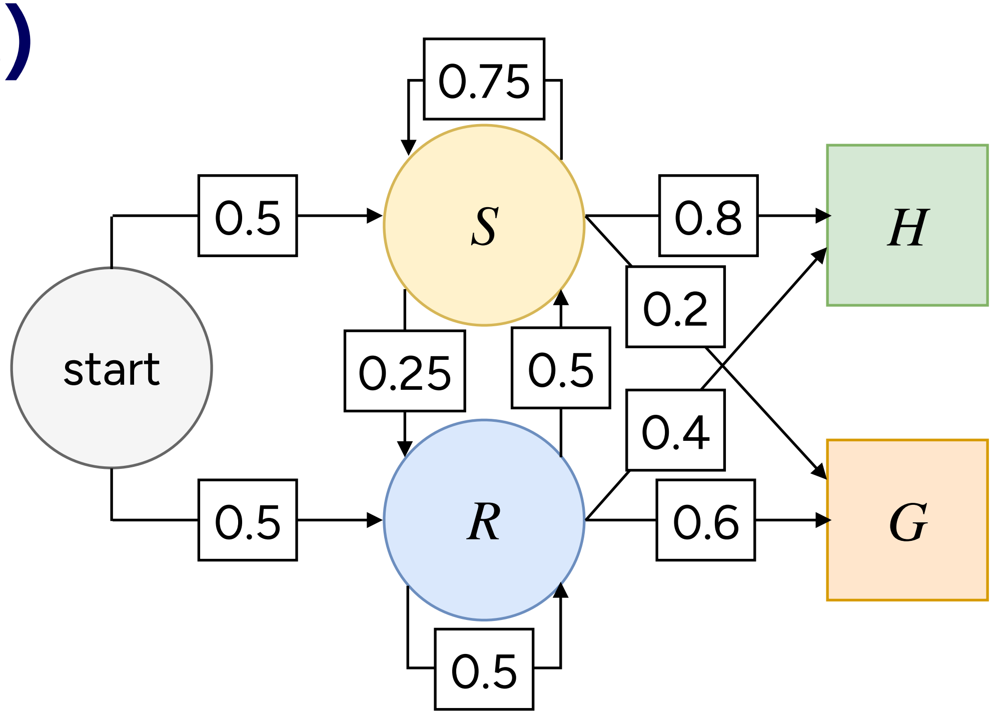
Trellis Source Code → Trellis Parser (1) → Trellis Model Compiler (2) → Set Constraint Validator (3) → Predecessor Analyzer (4) → CUDA Compiler (5) → Python Builder (6) → Python Library

- Introduction
- **Hidden Markov Models**
- Trellis
- Results
- Conclusion

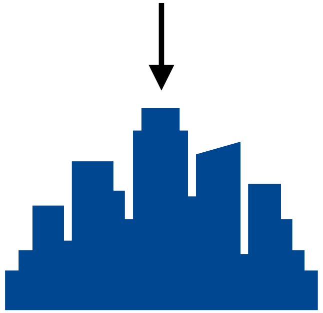
Hidden Markov Models

Hidden Markov Models (HMMs)

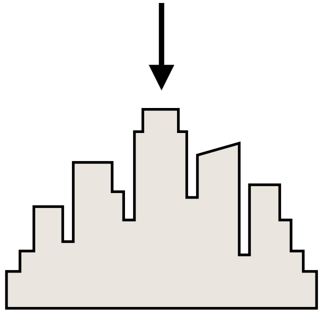
- Hidden states (S = sunny, R = rainy)
- Outputs (H = happy, G = grumpy)
- Initial probability
- Output probability
- Transition probability



Alice



Bob



- Introduction
- Hidden Markov Models
- **Trellis**
- Results
- Conclusion

Trellis

Trellis Language

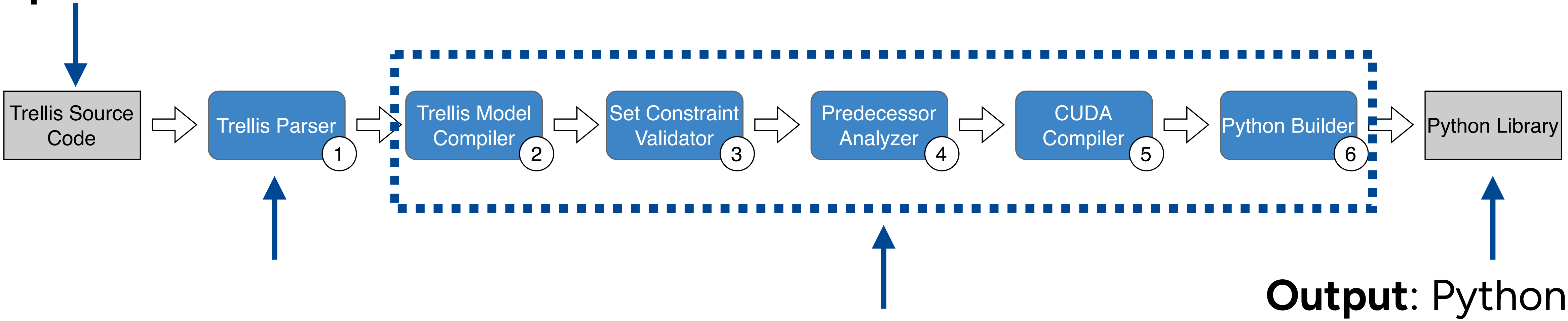
- Declarative HMM language
- Sparse model example
- Transition cases

Transition probability

```
let merlength = 3
let maxduration = 16
type Nucleotide = {A, C, G, T}
alias Kmer = Nucleotide[merlength]
alias Duration = 1 .. maxduration
alias ObsType = 0 .. 100
model {
 state = (Duration, Kmer)
 output = ObsType
 table initialProb(Duration, Kmer)
 table outputProb(ObsType, Kmer)
 table trans1(Kmer, Nucleotide)
 table trans2(Duration)
 table gamma()
 P(initial x) = initialProb(x[0], x[1])
 P(output o | x) = outputProb(o, x[1])
 P(transition x y) = {
   | { (1, [a, as...]) -> (k, [bs..., b]) | as == bs } =>
     trans1(x[1], y[1][2]) * trans2(y[0])
   | { (n1, x1) -> (n2, x2) | x1 == x2, n1 == maxduration, n2 == maxduration } =>
     gamma()
   | { (n1, x1) -> (n2, x2) | x1 == x2, n1 == maxduration, n2 == maxduration - 1 } =>
     1.0 - gamma()
   | { (n1, x1) -> (n2, x2) | x1 == x2, n2 == n1 - 1, n2 != maxduration - 1 } =>
     1.0
 }
}
```

Compiler Overview

Input: Trellis model



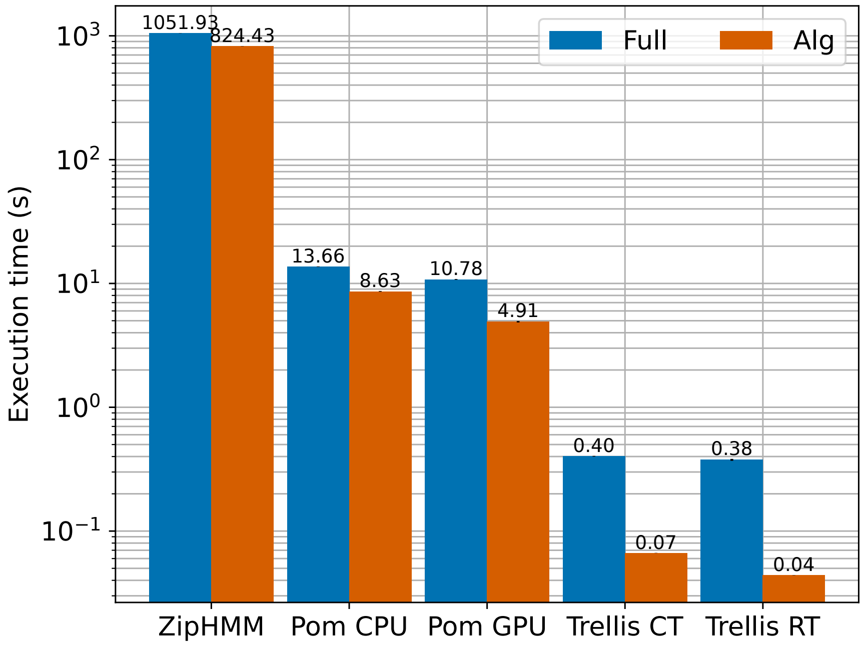
Output: Python library performing GPU calls

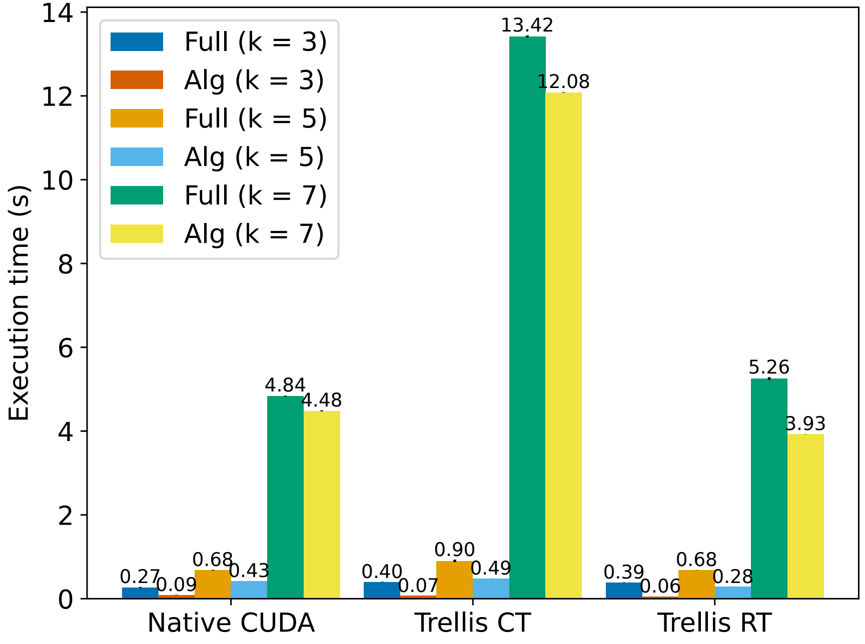
- Introduction
- Hidden Markov Models
- Trellis
- **Results**
- Conclusion

Results

Results

- Trellis outperforms previous work
- Matches performance of hand-written CUDA





- Introduction
- Hidden Markov Models
- Trellis
- Results
- **Conclusion**

Conclusion

Conclusion

- Trellis is a DSL designed for sparse HMMs
- Implemented in Miking

https://github.com/miking-lang/trellis-dsl

Thank you for listening!

References

- Sand, A., Kristiansen, M., Pedersen, C. N., & Mailund, T. (2013). zipHMMlib: a highly optimised HMM library exploiting repetitions in the input to speed up the forward algorithm. *BMC bioinformatics*, *14*, 1-11.
- Lunter, G. (2007). HMMoC—a compiler for hidden Markov models. *Bioinformatics*, *23*(18), 2485-2487.
- Schreiber, J. (2018). Pomegranate: fast and flexible probabilistic modeling in python. *Journal of Machine Learning Research*, *18*(164), 1-6.
- Lott, P. C., & Korf, I. (2014). StochHMM: a flexible hidden Markov model tool and C++ library. *Bioinformatics*, *30*(11), 1625-1626.
- Lam, T. Y., & Meyer, I. M. (2009). HMMCONVERTER 1.0: a toolbox for hidden Markov models. *Nucleic acids research*, *37*(21), e139-e139.