Grammatical Approaches to Problems in RNA Bioinformatics

Christian Höner zu Siederdissen

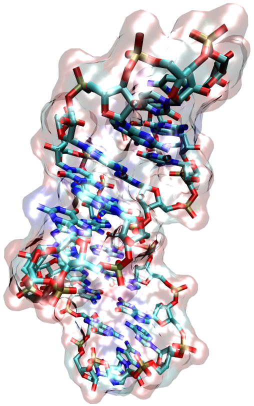
Institute for Theoretical Chemistry
University of Vienna

July, 23rd, 2013

Der Wissenschaftsfonds.

RNA Bioinformatics

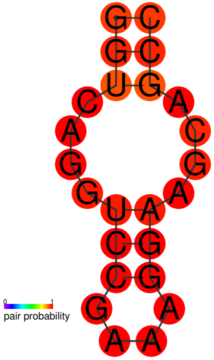
GGCUCUGUUUACCA<u>GGUCAGGUCCGAAAGGAAGCAGCC</u>AAGGCAGAGCC

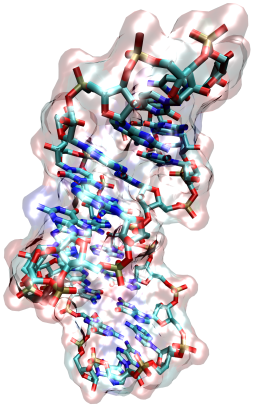


PDB: 1DUL

RNA Bioinformatics

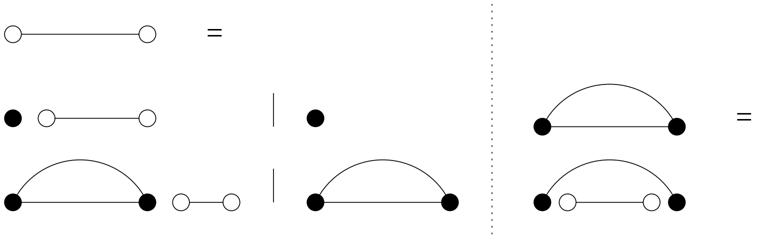
GGCUCUGUUUACCAGGUCAGGUCCGAAAGGAAGCAGCCAAGGCAGAGCC





PDB: 1DUL

Canonical RNA Folding



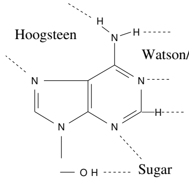
$$\begin{aligned} \mathit{Struct} \rightarrow \quad & \textsf{nuc} \quad \mathit{Struct} \\ | \quad & \textsf{nuc} \\ | \quad & \mathit{Pair} \quad \mathit{Struct} \\ | \quad & \mathit{Pair} \end{aligned}$$

$$\mathit{Pair} \rightarrow \quad \textsf{nuc} \quad \mathit{Struct} \quad \textsf{nuc}$$

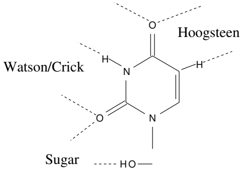
`based on the Vienna RNAfold package`

The Leontis-Westhof Notation

purine: adenine (A)



pyrimidine: uracil (U)



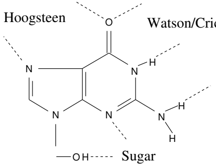
A,C,G,U

×

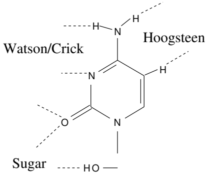
A,C,G,U

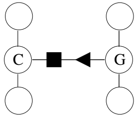
●	Watson-Crick (cis/anti-parallel)
□	Hoogsteen (trans/parallel)
▷	Sugar

purine: guanine (G)

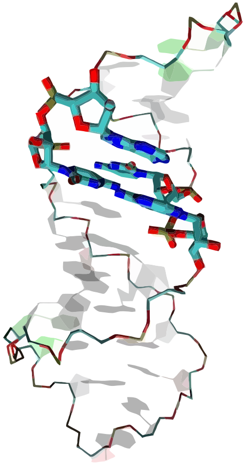


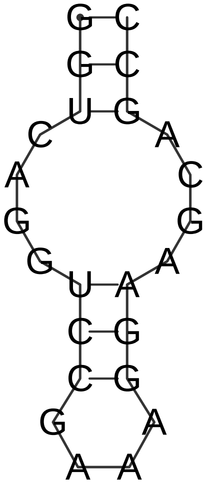
pyrimidine: cytosine (C)





Extended RNA secondary Structures

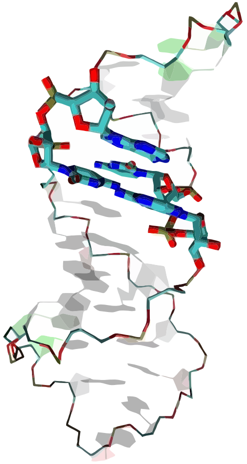


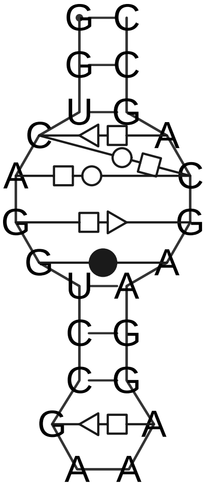


●	Watson-Crick (cis/anti-parallel)
□	Hoogsteen (trans/parallel)
▷	Sugar

- allow full Leontis-Westhof annotation
- shared nucleotides
- interior loops are filled with non-WC basepairs

Extended RNA secondary Structures

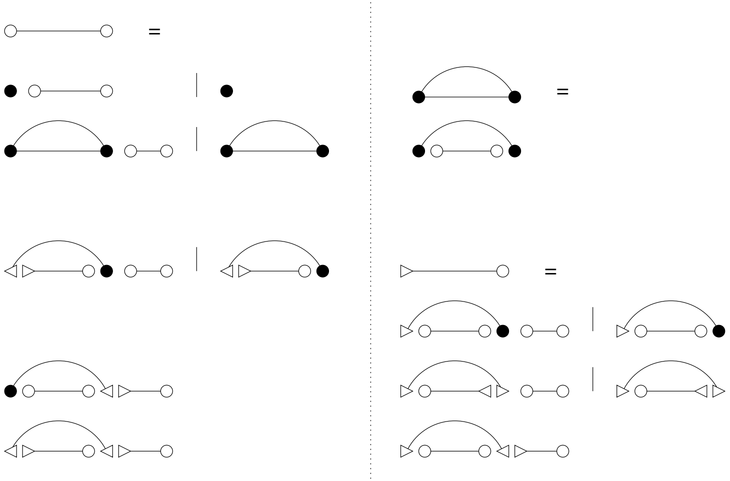




●	Watson-Crick (cis/anti-parallel)
□	Hoogsteen (trans/parallel)
▷	Sugar

- allow full Leontis-Westhof annotation
- shared nucleotides
- interior loops are filled with non-WC basepairs

Extended RNA Folding



`full grammar in appendix`

Training Data

- canonical base pairs only:

melting experiments	short sequences, reliable, free energy measurements
RNAstrand	curated, larger number of long sequences
Rfam	partially curated, many predicted consensus secondary structures

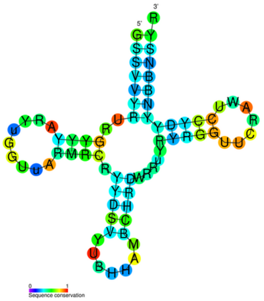
- extended base pair information:

PDB	small set of sequences, detailed base pair information

Results

program	count	mcc	f-measure	sensitivity	ppv
RNAwolf, ≤ 150	250	0.62	0.62	0.67	0.59
RNAwolf, > 150	250	0.45	0.44	0.57	0.37
RNAwolf, ext. pairs	300	0.34	0.34	0.34	0.34
RNAwolf, non-cWW	300	0.46	0.46	0.48	0.45
RNAfold v1.8.5	500	0.67	0.67	0.71	0.65
BL*	500	0.71	0.71	0.74	0.70

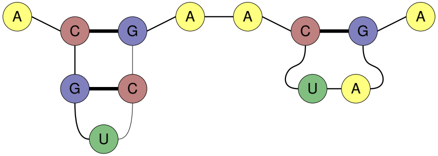
Non-coding RNA Search

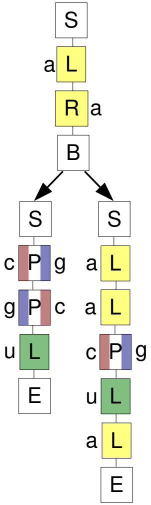


$$\begin{tikzcd}
\text{Genome} \arrow[r] \arrow[dr] & \text{RNA} \arrow[r] & \text{Protein} \arrow[ddl, "\text{Searching with HMMs (HMMER3)}"] \\
& \text{Structural ncRNA} \arrow[d, "\text{Searching with SCFGs (Infernal)}"'] & \\
& \text{Other Genomes} &
\end{tikzcd}$$

(Infernal) RNA Family Models

```
human     acgucg aacuaga
cow       accugg aacuaga
dog       acuugg aag uca
cat       acgucgaaacuaga
structure *<<*>>.**<**>*
```





Syntactic vs. Semantic Ambiguity in RNA Folding

Syntactic different parses on the same sequence produce different objects (wanted)

```
gcaagc   ((..))   (....)   .(..).   ......
          0.5      0.2      0.2      0.1
```

Semantic different parses on the same sequence produce the same object (unwanted)

```
gcaagc   ((..))   (....)   ((..))   ((..)) etc
          0.15      0.2     0.15     0.05
```

Semantics of Family Models

sequence alignment recap:

```
ACAGGGG---CAC   ACA---GGGGCAC     ACA[GGGG]CAC
ACA----TTTCAC   ACATTT----CAC     ACA[TTT] CAC
```

three meaningful semantics can be defined for family models:

```
 Consensus **<<**>>

 Alignment **<<*--*>>     **<<*--*>>     **<<**>>
           __((....))     .._(....)_     ..((..))

     Trace *-*<<**>>     **<-<**>>
           ..__(..)_     .__.(..)_
           allowed       banned

Structural **<<*-*>>      consensus implicit only
             ((...))
```

Non-ambiguous Trace Semantics for Family Models

```
remember:     ACA[GGGG]CAC
              ACA[TTT] CAC
```

$$
\begin{array}{lcl}
A & \rightarrow & \genfrac{}{}{0pt}{}{-}{.}\ A \mid M \\
M & \rightarrow & \varepsilon \mid \genfrac{}{}{0pt}{}{*}{.}\ A \mid \underline{*}\ M \mid \\
 & & \genfrac{}{}{0pt}{}{<}{(}\ A\ \genfrac{}{}{0pt}{}{>}{)}\ A \mid \genfrac{}{}{0pt}{}{<}{.}\ A\ \underline{>}\ M \mid \\
 & & \underline{<}\ M\ \genfrac{}{}{0pt}{}{>}{.}\ A \mid \underline{<}\ M\ \underline{>}\ M
\end{array}
$$

- proved unambiguous using the ACLA ambiguity checker
- by virtue of construction, the above grammar generates unambiguous model grammars

Results: Counting Alignments and Traces

Model length (size)	RF00163 45 (31)	RF01380 19 (12)
$\lvert x \rvert = 12$		
structures	8,958	2,048
traces	35×10^9	141,120,525
alignments	715×10^{12}	35,330,137,025
$\lvert x \rvert = 31$		
structures	n.a.	n.a.
traces	2×10^{21}	30,405,943,383,200
alignments	2×10^{27}	208,217,738,981,165,823

RF00163 consensus:

<<<<<<*******<<<<******>>>>***<<<<>>>>*>>>>>>

RF01380 consensus:

<<<<<<<****>>>*>>>>

Grammars in RNA Bioinformatics

- well-established, formal language
- high-level view (no indices!)
- efficient implementations (same asymptotics, similar constants)
- separation of concerns (search space, evaluation, implementation)

Future Developments

- multi-tape problems
- heterogeneous and partially ordered index spaces
- Products of Grammars
- training with L_1 regularization (for all grammars)
- fine- and coarse-grained parallelism

Acknowledgments

Vienna	Ivo Hofacker
Leipzig	Stephan Bernhart, Peter Stadler
Bielefeld	Robert Giegerich
Vie/Lei	everybody at the TBI & BioInf Leipzig
⋋	the Haskell community

universität wien

FWF Der Wissenschaftsfonds.

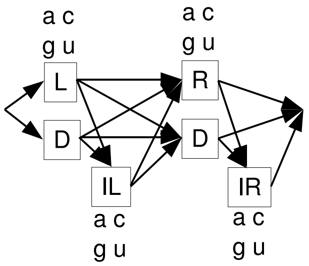
funding by the Austrian FWF:
GEN-AU projects: "BIN III" and "regulatory ncRNAs"
SFB F43 "RNA Regulation of the transcriptome"

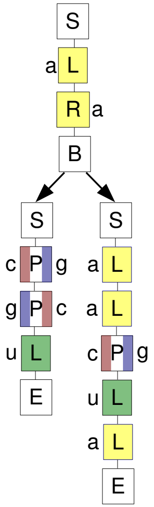
Publications in Thesis

- Giegerich, Robert and Christian Höner zu Siederdissen (2011). "Semantics and Ambiguity of Stochastic RNA Family Models". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 8.2, pp. 499–516.
- Höner zu Siederdissen, Christian and Ivo L. Hofacker (2010). "Discriminatory power of RNA family models". In: *Bioinformatics* 26.18, pp. 453–459.
- Höner zu Siederdissen, Christian, Stephan H. Bernhart, et al. (2011). "A folding algorithm for extended RNA secondary structures". In: *Bioinformatics* 27.13, pp. 129–136.
- Höner zu Siederdissen, Christian (2012). "Sneaking Around concatMap: Efficient Combinators for Dynamic Programming". In: *Proceedings of the 17th ACM SIGPLAN international conference on Functional programming*. ICFP '12. Copenhagen, Denmark: ACM, pp. 215–226. ISBN: 978-1-4503-1054-3.

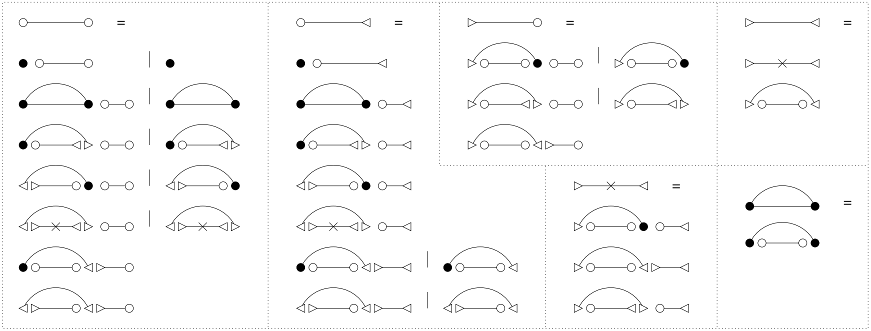
(Infernal) RNA Family Models

```
human     acgucg aacuaga
cow       accugg aacuaga
dog       acuugg aag uca
cat       acgucgaaacuaga
structure *<<*>>.**<**>*
```





Extended RNA Folding: Complete Grammar



- $O(\alpha \times 3 \times n^3 + \beta \times 400 \times n^2)$ runtime
- $O(\{3^2 \times 2\} \times 10 \times n^2)$ space
- 6 non-terminals and additional helper tables
- interior loop closing pairs assumed independent

Parameter Training

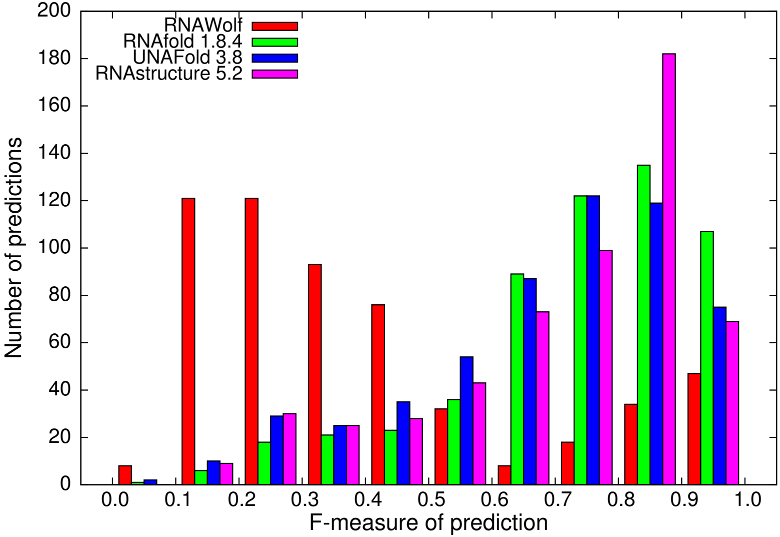
- melting energy: y, melting structural features: A
- structural constraints (known - predicted): D
  energy difference: d
- generate constraints iteratively (cf. Andronescu et al, 2007)
- destabilizing features (hairpins, bulges, interior loops): S

$$\left\| \begin{pmatrix} A & 0 \\ D & -I \end{pmatrix} \begin{pmatrix} x_{\text{cur}} \\ d_{\text{init}} \end{pmatrix} - \begin{pmatrix} y \\ d \end{pmatrix} \right\|_2 \quad + \lambda \left\| x \right\|_1$$

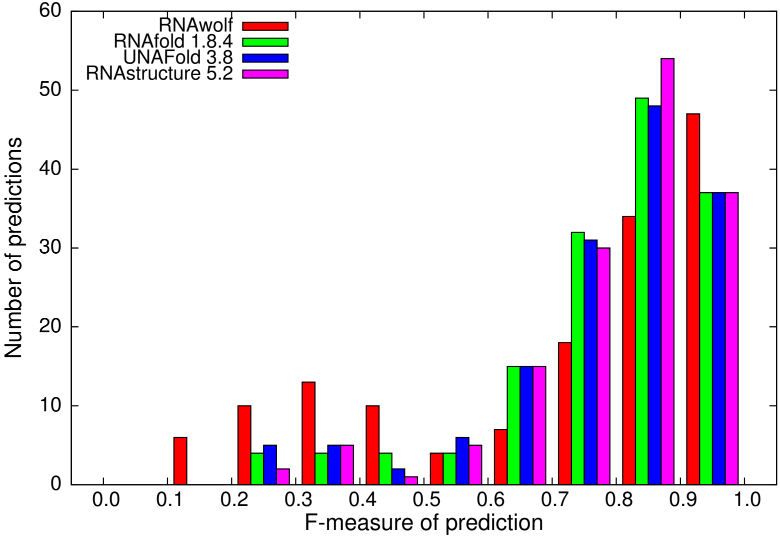
with linear constraints

$$-5 < x_j < 5, \qquad 0 < x_m, \quad m \in S, \qquad 0 < d_k$$

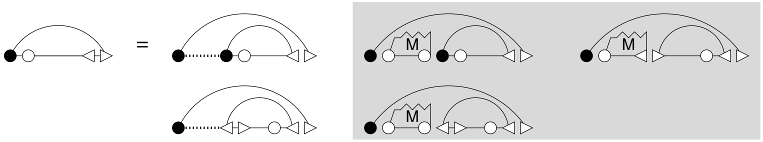
Results: complete set



Results: PDB only



Multibranched Loops



Base Pair Probabilities in the PDB

cWW						tSH	tHS	tsS
G-C	C-G	U-A	A-U	G-U	U-G	G-A	A-G	G-A
73 342	68 083	23 606	23 419	10 168	9 644	7 742	6 798	5 121
0.249	0.231	0.080	0.079	0.035	0.033	0.026	0.023	0.017

tWH	csS	tSs	tHW	cSs	csS	cSH	cSs	Rest
U-A	C-A	A-G	A-U	C-A	A-C	G-U	A-C	
4 474	3 638	2 863	2 851	2 564	2 109	2 072	1 917	44 302
0.015	0.012	0.010	0.010	0.009	0.007	0.007	0.007	0.150

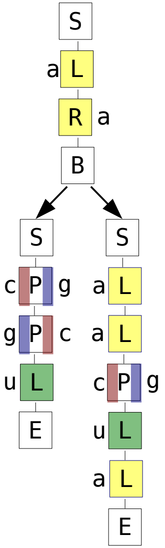
Base Pair Types

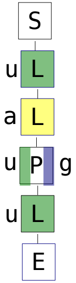
pair type	base pairs	base triplets	base quadruplets	base quintets (?)
number	261 842	15 288	761	3 (?)
fraction	0.942	0.055	0.003	–

The *Link Score*

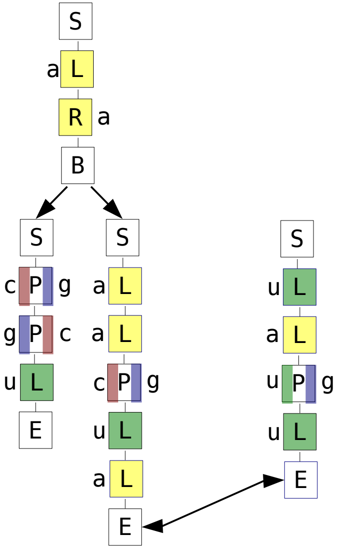
- take all sequences [acgt]*:
  ϵ ... acagtgctagtcagtcgatcgatcgatcgatc
- take two CMs: M_1 and M_2:
  - $M_1(\epsilon) \rightarrow score = -5$
    $M_2(\epsilon) \rightarrow score = -5$
  - $M_1(\text{acagt}\ldots) \rightarrow score = 15$
    $M_2(\text{acagt}\ldots) \rightarrow \underline{score = 10}$
  - *Link Score*: 10
- for each sequence, take the smaller of the two scores
- for all scores, take the largest: the *Link Score*

DP on two Input CMs

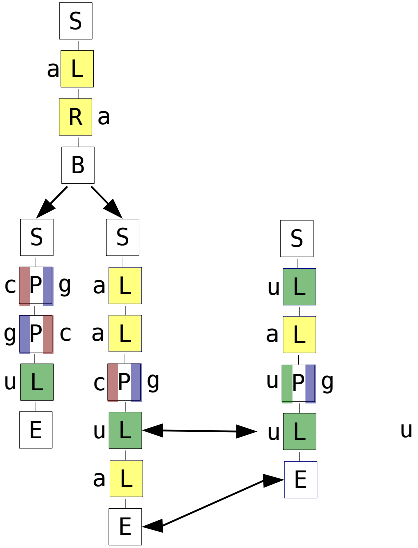




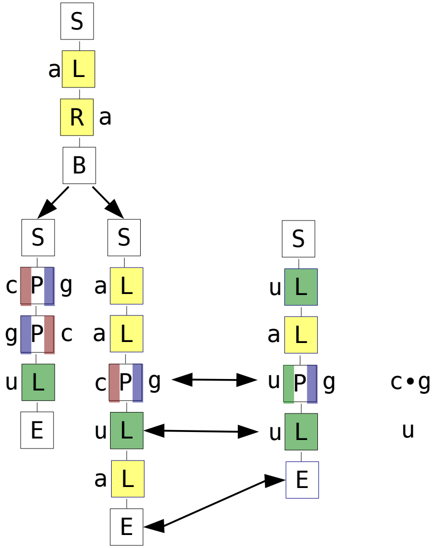
DP on two Input CMs



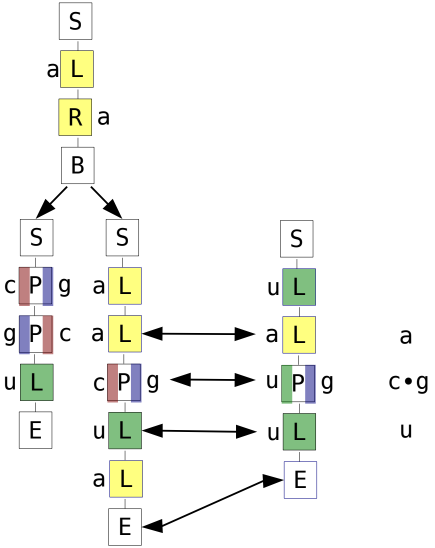
DP on two Input CMs



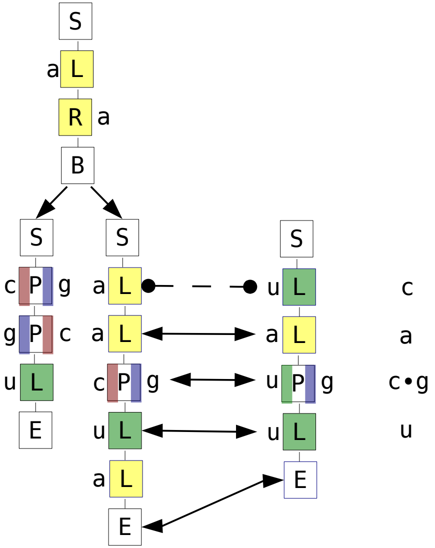
DP on two Input CMs



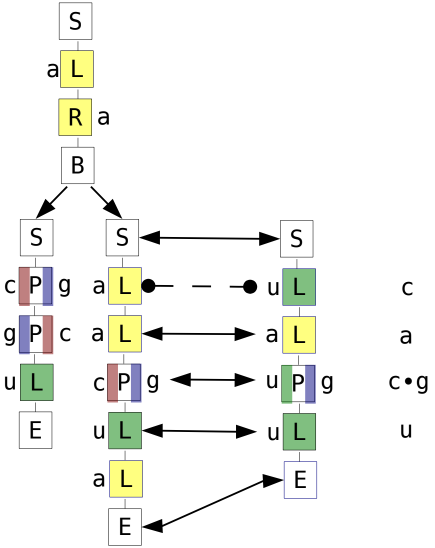
DP on two Input CMs



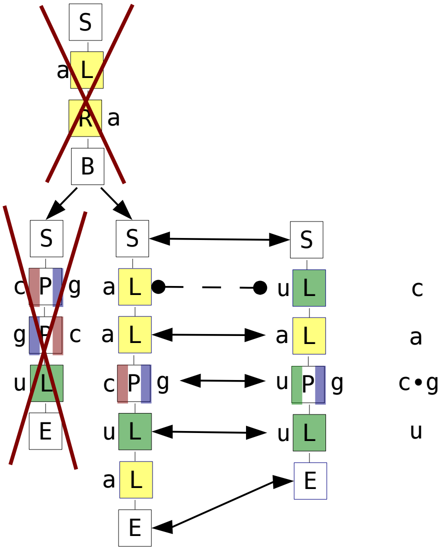
DP on two Input CMs



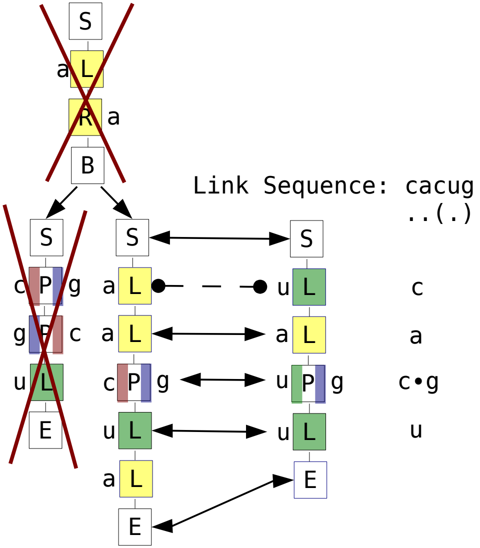
DP on two Input CMs



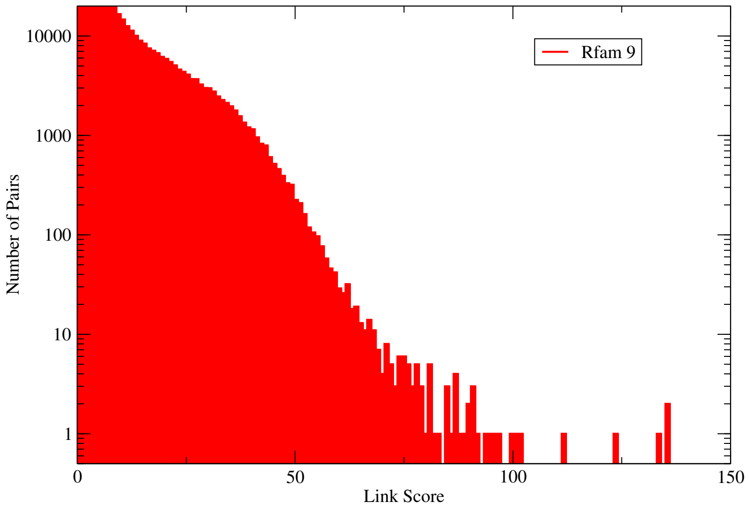
DP on two Input CMs



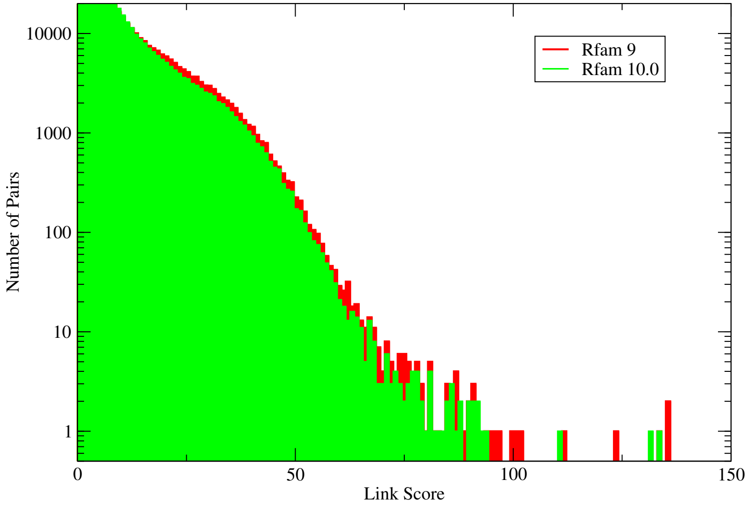
DP on two Input CMs



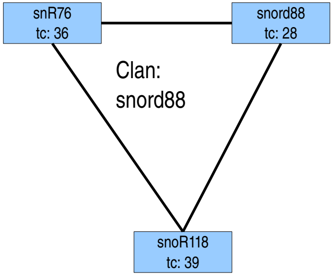
Pairs of Families with Overlap



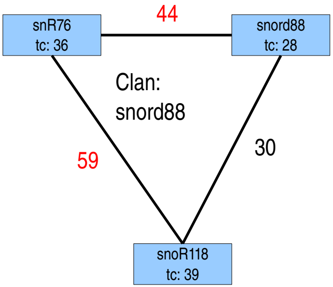
Pairs of Families with Overlap



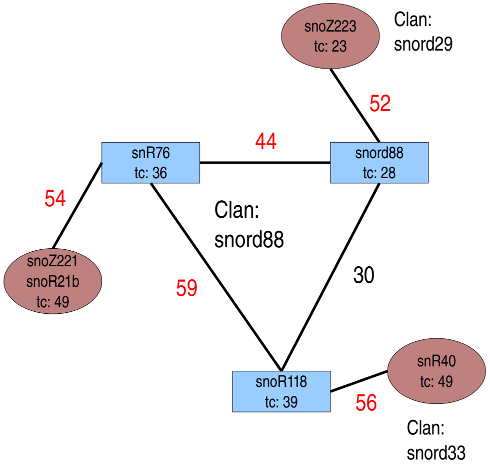
Clan snord88



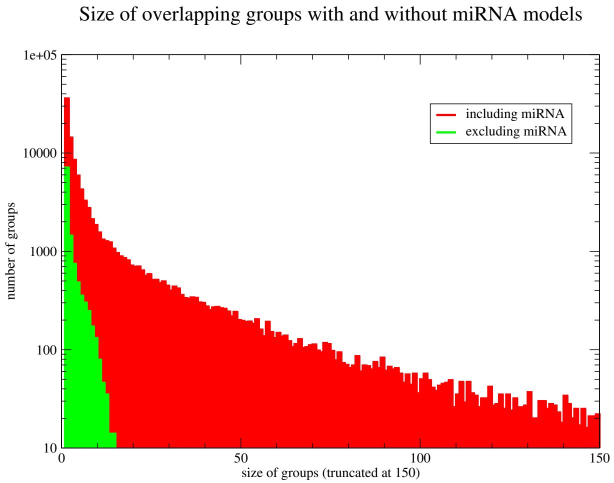
Clan snord88



Clan snord88

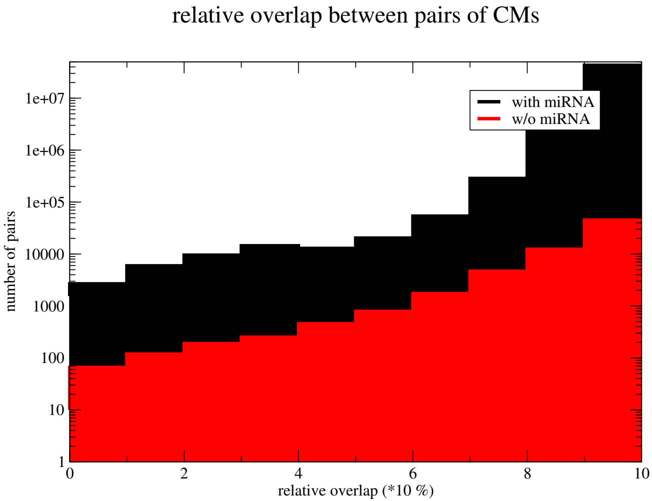


Real World Data



(with Jan Gorodkin and coworkers, RTH, Copenhagen)

Real World Data

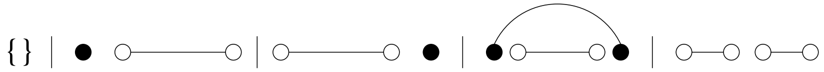


(with Jan Gorodkin and coworkers)

$$\mathrm{MaxiMin}\quad (k_{\mathbf{1}}, k_{\mathbf{2}}) =$$

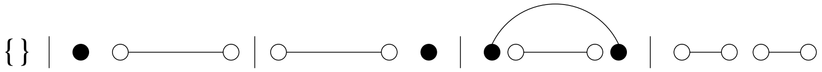
$$\begin{cases}
(0,0) & k_{\mathbf{1}} = \mathsf{E} \wedge k_{\mathbf{2}} = \mathsf{E} \\
\mathrm{maxmin}\{\mathrm{MaxiMin}(k'_{\mathbf{1}}, k'_{\mathbf{2}}) + (e_{k_{\mathbf{1}},a,b}, e_{k_{\mathbf{2}},a,b}) + (t_{k_{\mathbf{1}}\to k'_{\mathbf{1}}}, t_{k_{\mathbf{2}}\to k'_{\mathbf{2}}}) & \\
\quad \mid \quad k'_{\mathbf{1}} \in c_{k_{\mathbf{1}}}, k'_{\mathbf{2}} \in c_{k_{\mathbf{2}}}, a \in \mathcal{A}, b \in \mathcal{A}\} & k_{\mathbf{1}} = \mathsf{P} \wedge k_{\mathbf{2}} = \mathsf{P} \\
\mathrm{maxmin}\{\mathrm{MaxiMin}(k'_{\mathbf{1}}, k'_{\mathbf{2}}) + (e_{k_{\mathbf{1}},a}, e_{k_{\mathbf{2}},a}) + (t_{k_{\mathbf{1}}\to k'_{\mathbf{1}}}, t_{k_{\mathbf{2}}\to k'_{\mathbf{2}}}) & \\
\quad \mid \quad k'_{\mathbf{1}} \in c_{k_{\mathbf{1}}}, k'_{\mathbf{2}} \in c_{k_{\mathbf{2}}}, a \in \mathcal{A}\} & k_{\mathbf{1}} \in \{\mathsf{L},\mathsf{IL}\} \wedge k_{\mathbf{2}} \in \{\mathsf{L},\mathsf{IL}\} \\
\mathrm{maxmin}\{\mathrm{MaxiMin}(k'_{\mathbf{1}}, k'_{\mathbf{2}}) + (e_{k_{\mathbf{1}},b}, e_{k_{\mathbf{2}},b}) + (t_{k_{\mathbf{1}}\to k'_{\mathbf{1}}}, t_{k_{\mathbf{2}}\to k'_{\mathbf{2}}}) & \\
\quad \mid \quad k'_{\mathbf{1}} \in c_{k_{\mathbf{1}}}, k'_{\mathbf{2}} \in c_{k_{\mathbf{2}}}, b \in \mathcal{A}\} & k_{\mathbf{1}} \in \{\mathsf{R},\mathsf{IR}\} \wedge k_{\mathbf{2}} \in \{\mathsf{R},\mathsf{IR}\} \\
\mathrm{maxmin}\{\mathrm{MaxiMin}(k_{\mathbf{1}}, k'_{\mathbf{2}}) + (0, t_{k_{\mathbf{2}}\to k'_{\mathbf{2}}}) & \\
\quad \mid \quad k'_{\mathbf{2}} \in c_{k_{\mathbf{2}}}\} & k_{\mathbf{1}} = \mathsf{E} \wedge k_{\mathbf{2}} \in \{\mathsf{D},\mathsf{S}\} \\
\mathrm{maxmin}\{\mathrm{MaxiMin}(k'_{\mathbf{1}}, k_{\mathbf{2}}) + (t_{k_{\mathbf{1}}\to k'_{\mathbf{1}}}, 0) & \\
\quad \mid \quad k'_{\mathbf{1}} \in c_{k_{\mathbf{1}}}\} & k_{\mathbf{1}} \in \{\mathsf{D},\mathsf{S}\} \wedge k_{\mathbf{2}} = \mathsf{E} \\
\mathrm{maxmin}\{\{\mathrm{MaxiMin}(k'_{\mathbf{1},\mathbf{1}}, k'_{\mathbf{2},\mathbf{1}}) + \mathrm{MaxiMin}(k'_{\mathbf{1},\mathbf{2}}, k'_{\mathbf{2},\mathbf{2}}) & \\
\quad \mid \quad \{k'_{\mathbf{1},\mathbf{1}}, k'_{\mathbf{1},\mathbf{2}}\} = c_{k_{\mathbf{1}}}, \{k'_{\mathbf{2},\mathbf{1}}, k'_{\mathbf{2},\mathbf{2}}\} = c_{k_{\mathbf{2}}}\} \quad \cup & \\
\qquad \{\mathrm{MaxiMin}(k'_{\mathbf{1},\mathbf{2}}, k'_{\mathbf{2},\mathbf{1}}) + \mathrm{MaxiMin}(k'_{\mathbf{1},\mathbf{1}}, \mathsf{E}) + \mathrm{MaxiMin}(\mathsf{E}, k'_{\mathbf{2},\mathbf{2}}) & \\
\quad \mid \quad \{k'_{\mathbf{1},\mathbf{1}}, k'_{\mathbf{1},\mathbf{2}}\} = c_{k_{\mathbf{1}}}, \{k'_{\mathbf{2},\mathbf{1}}, k'_{\mathbf{2},\mathbf{2}}\} = c_{k_{\mathbf{2}}}\} \quad \cup & \\
\qquad \{\mathrm{MaxiMin}(k'_{\mathbf{1},\mathbf{1}}, k'_{\mathbf{2},\mathbf{2}}) + \mathrm{MaxiMin}(k'_{\mathbf{1},\mathbf{2}}, \mathsf{E}) + \mathrm{MaxiMin}(\mathsf{E}, k'_{\mathbf{2},\mathbf{1}}) & \\
\quad \mid \quad \{k'_{\mathbf{1},\mathbf{1}}, k'_{\mathbf{1},\mathbf{2}}\} = c_{k_{\mathbf{1}}}, \{k'_{\mathbf{2},\mathbf{1}}, k'_{\mathbf{2},\mathbf{2}}\} = c_{k_{\mathbf{2}}}\}\} & k_{\mathbf{1}} = \mathsf{B} \wedge k_{\mathbf{2}} = \mathsf{B} \\
\mathrm{maxmin}\{\mathrm{MaxiMin}(k'_{\mathbf{1},\mathbf{1}}, k_{\mathbf{2}}) + \mathrm{MaxiMin}(k'_{\mathbf{1},\mathbf{2}}, \mathsf{E}) & \\
\quad \mid \quad \{k'_{\mathbf{1},\mathbf{1}}, k'_{\mathbf{1},\mathbf{2}}\} = c_{k_{\mathbf{1}}}\} & k_{\mathbf{1}} = \mathsf{B} \wedge k_{\mathbf{2}} \neq \mathsf{B} \\
\mathrm{maxmin}\{\mathrm{MaxiMin}(k'_{\mathbf{1}}, k'_{\mathbf{2}}) + (t_{k_{\mathbf{1}}\to k'_{\mathbf{1}}}, t_{k_{\mathbf{2}}\to k'_{\mathbf{2}}}) & \\
\quad \mid \quad k'_{\mathbf{1}} \in c_{k_{\mathbf{1}}}, k'_{\mathbf{2}} \in c_{k_{\mathbf{2}}}\} & (k_{\mathbf{1}}, k_{\mathbf{2}}) \in \{(\mathsf{S},\mathsf{S}),(\mathsf{D},\mathsf{D})\} \\
(-\infty, -\infty) & \text{otherwise}
\end{cases}$$

A Simple Grammar: Nussinov78



```
S → ϵ          -- terminate on empty input
S → a S        -- read a single character to the left
S →   S b      -- read a single character to the right
S → a S b      -- read two bracketing characters
S →  S S       -- split the input
```

A Simple Grammar: Nussinov78



```
(s, (
  nil    ⋘  empty                  |||
  left   ⋘  base  %  s             |||
  right  ⋘           s  %  base    |||
  pair   ⋘  base  %  s  %  base    |||
  split  ⋘       s'  %  s'         ... h
) where s' = nonEmpty s )
```

Algebra

```
data Signature = Signature
{  nil   :: e → S
,  left  :: A → S → S
,  right :: S → A → S
,  pair  :: A → S → A → S
,  split :: S → S → S
,  h     :: Stream S → S }

pairmax = Signature
{   nil   = λ _      → 0
,   left  = λ _ x    → x
,   right = λ   x _  → x
,   pair  = λ l x r  → if isPair l r then x + 1 else x
,   split = λ  x y   → x + y
,   h     = λ xs     → maximumS xs }
```

Algebra

```
data Signature = Signature
{  nil   :: e → S
,  left  :: A → S → S
,  right :: S → A → S
,  pair  :: A → S → A → S
,  split :: S → S → S
,  h     :: Stream S → S }

pretty = Signature
{   nil   = λ _     → ""
,   left  = λ _ x   → "." ++ x
,   right = λ   x _ → x ++ "."
,   pair  = λ l x r → "(" ++ x ++ ")"
,   split = λ  x y  → x ++ y
,   h     = λ xs    → xs }
```

Algebra Products & Backtracking

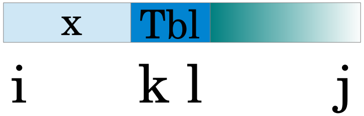
`pairmax <** pretty`

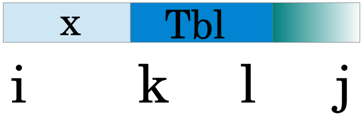
- algebra products provide convenient capability extensions
- `pairmax` yields the optimal score
- `prettyprint` prints a parse
- `pairmx <** prettyprint` prints the parses for the optimal score
- `class *** pairmax` allows for classified dynamic programming

- `(<**), (***)` need to be defined for each grammar
- use TemplateHaskell for automation

```
(<**) f s = STwoWay l_s s_l s_s n_n h where
  STwoWay lsf slf ssf nnf hf = f
  STwoWay lss sls sss nns hs = s
  l_s = go lsf lss
  s_l = go slf sls
  s_s = go ssf sss
  n_n e = (nnf e, return $ S.singleton $ nns e)
  h xs = do
    hfs ← hf $ S.map fst xs
    let phfs = S.concatMapM snd
              ∘ S.filter ((hfs==) ∘ fst) $ xs
    hs phfs
  go funL funR (x,ys) c = (funL x c, ys >>=
            return ∘ S.map (λy → funR y c))
```

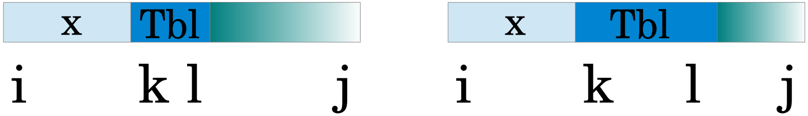
mkStream on Outer Elements





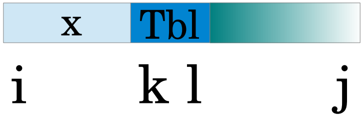
```
mkStream (ls :!: s'@(Table tbl)) Outer !ij@(i:.j)
  = S.map (λs → let (Subword (_:.l)) = getIdx s
                in ElmT s (tbl!(l:.j)) (l:.j)
          )
  $ mkStream ls (Inner Check) (i:.j)
```

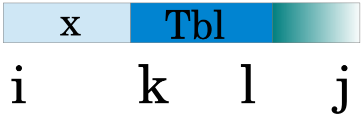
mkStream on Inner Elements



```
mkStream (ls :!: s'@(Table tbl)) (Inner _) ij@(i:.j)
  = S.flatten mk step $ mkStream ls (Inner NoCheck) ij where
  mk s = let (_:.k) = getIdx s
         in return (s :!: k :!: k)
  step (s :!: k :!: l)
     | l > j       = S.Done
     | otherwise = S.Yield (ElmT s (tbl!(k:.l)) (k:.l))
                           (s :!: k :!: l+1)
```

mkStream on Z

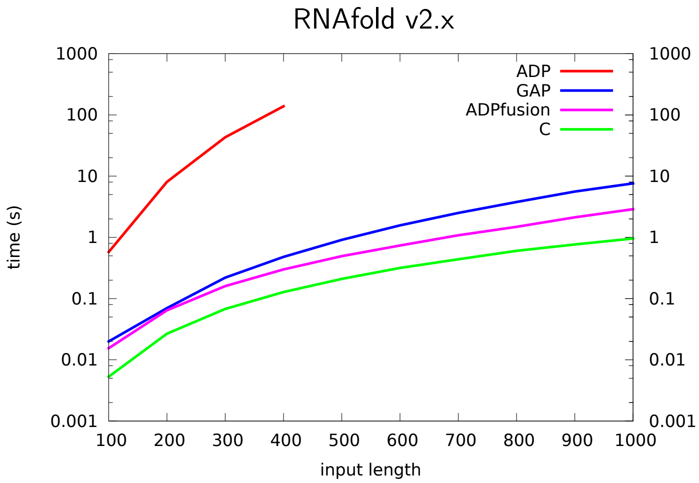




```
mkStream Z (Inner NoCheck) (i:.j)
  = S.singleton $ ElmZ (i:.i)


mkStream Z (Inner Check)   (i:.j)
  = S.unfoldr step i where
    step !k
      | k≤j       = P.Just $ (ElmZ (i:.i), j+1)
      | otherwise = P.Nothing
```

The Obligatory Benchmark Slide



Multi-Dimensional Grammars

	G	L	O	B	A	L
L	-1	0	-5	-7	-9	-8
O	-3	-2	2	0	-2	-4
C	-5	-4	0	1	-1	-3
A	-7	-6	-2	-1	3	-2
L	-9	-5	-4	-3	-2	5

```
G   L   O   B   A   L
--  L   O   C   A   L
```

$$
\begin{aligned}
X_{ij} = \mathrm{opt}\{ & X_{i-1,j-1} \\
& + \delta(x_i, y_j), X_{i-1,j} \\
& + \beta(x_i), X_{i,j-1} \\
& + \beta(y_j), \epsilon_{i=j=0}\}
\end{aligned}
$$

```
( x, step_step  ⋘  x % (T :. c    :. c   )  |||
     step_loop  ⋘  x % (T :. c    :. None)  |||
     loop_step  ⋘  x % (T :. None :. c   )  |||
     nil_nil    ⋘      (T:.Empty:.Empty)   ... h )
```

- multi-dim grammars use the same framework
- non-terminals, algebras, production rules are the same
- *terminal* symbols are now multi-dimensional

Das Kleine 1 × 1 der Grammatiken

Too lazy to write a complex DP algorithm? What you want to write has some structure? try this:

```
  [qqGrammar|
Grammar: Step
NT: W
T:  c
W → step  ⋘  W c
W → loop  ⋘  W
//
Grammar: Done
NT: W
T:  empty
W → nil  ⋘  empty
//
Grammar: Loop
NT: W
W → loop  ⋘  W
//
Product:  TwoWay
Step ⋊⋉ Step + Done ∗
2 - Loop ∗ 2
//
|]
```

Das Kleine 1×1 der Grammatiken

- algebraic framework that formalizes how to multiply dynamic programming algorithms
- currently for linear grammars
- context-free grammars require some additional thoughts

- directly embedded in Haskell via QuasiQuoting/TemplateHaskell
- small, extensible DSL
- user-extensible parser and interpreter
- generates fast ADPfusion code
- scales to “all useful” dimensions

Christian Höner zu Siederdissen, Ivo L. Hofacker, Peter F. Stadler
How to Multiply Dynamic Programming Algorithms

Other Publications I

- Höner zu Siederdissen, Christian, Ivo L. Hofacker, and Peter F. Stadler (2013). "How to Multiply Dynamic Programming Algorithms". In: *Brazilian Symposium on Bioinformatics (BSB 2013)*. Vol. 8213. Lecture Notes in Bioinformatics. Springer, Heidelberg, pp. 82–93.
- Höner zu Siederdissen, Christian (2013). "`ADPfusion`: Efficient Dynamic Programming over Sequence Data". In: *HaL8 Workshop*.
- Höner zu Siederdissen, Christian, Stefan Hammer, et al. (2013). "Computational Design of RNAs with Complex Energy Landscapes". In: *Biopolymers* 99.12, pp. 1124–1136. ISSN: 1097-0282.
- Theis, Corinna et al. (2013). "Automated identification of 3D modules with discriminative power in RNA structural alignments". In: *Nucleic Acids Research* 41.22, pp. 9999–10009.
- Eggenhofer, Florian, Ivo L. Hofacker, and Christian Höner zu Siederdissen (2013). "CMCompare webserver: comparing RNA families via covariance models". In: *Nucleic Acids Research* 41.W1, W499–W503.

Other Publications II

- Lorenz, Ronny, Stephan H. Bernhart, Jing Qin, et al. (2013). "2D meets 4G: G-Quadruplexes in RNA Secondary Structure Prediction". In: *IEEE/ACM Transactions on Computation Biology and Bioinformatics*.
- Vierna, Joaquin et al. (2013). "Systematic analysis and evolution of 5S ribosomal DNA in metazoans". In: *Heredity* 111.5, pp. 410–421.
- Lorenz, Ronny, Stephan H. Bernhart, Fabian Externbrink, et al. (2012). "RNA Folding Algorithms with G-Quadruplexes". In: *Brazilian Symposium on Bioinformatics (BSB 2012)*. Ed. by M.C.P. De Souto and M.G. Kann. Vol. 7409. Lecture Notes in Bioinformatics. Springer, Heidelberg, pp. 49–60.
- Helm, Conrad et al. (2012). "Deep sequencing of small RNAs confirms an annelid affinity of Myzostomida". In: *Molecular Phylogenetics and Evolution* 64 (1), pp. 198–203.
- Lorenz, Ronny, Stephan H. Bernhart, Christian Höner zu Siederdissen, et al. (2011). "ViennaRNA Package 2.0". In: *Algorithms for Molecular Biology* 6.26.

Other Publications III

- Marz, Manja et al. (2011). "Animal snoRNAs and scaRNAs with exceptional structures". In: *RNA Biology* 8.6, pp. 1–9.
- Hackl, Matthias et al. (2011). "Next-generation sequencing of the Chinese hamster ovary microRNA transcriptome: identification, annotation and profiling of microRNAs as targets for cellular engineering". In: *Journal of Biotechnology* 153, pp. 62–75.
- Höner zu Siederdissen, Christian, Susanne Ragg, and Sven Rahmann (2007). "Discovering Biomarkers for Myocardial Infarction from SELDI-TOF Spectra". In: *Advances in Data Analysis*. Ed. by Reinhold Decker and Hans -J. Lenz. Studies in Classification, Data Analysis, and Knowledge Organization. Springer Berlin Heidelberg, pp. 569–576. ISBN: 978-3-540-70981-7.