Logol

Simple design for complex pattern

7ème journée de la plateforme

26/10/09

Olivier SALLOU

What is Logol ?

- A Grammar language to define biological patterns
- A software suite implementing the grammar
  - LogolDesigner, LogolMatch, LogolAnalyser

What is it for? 1/2

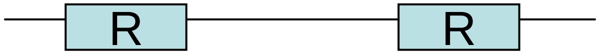
- Define some patterns, like in Perl language:
  - Example: (*w+_d+)[ac]* to match a specific word instance in a string
- Find patterns with a biological meaning for DNA/RNA/Proteins sequences
  - « I want to find *acgt* repeated 3 or 4 times ».
- Keep matches details in results within a coherent structure

What is it for? 2/2

- Find all exact matches
  - No false positive or missing result
  - Accepts errors/distance in model
- Find occurrences within an acceptable amount of time
- On a whole genome or on multiple sequences

Why a new pattern tool?

- Many existing tools focus on basic regular expressions description, e.g. find a word within a string
- Others focus on specialized patterns (loops,…)
- None or limited reuse of internal variables
  - Example:
    - I want to find a repetition (LTR):

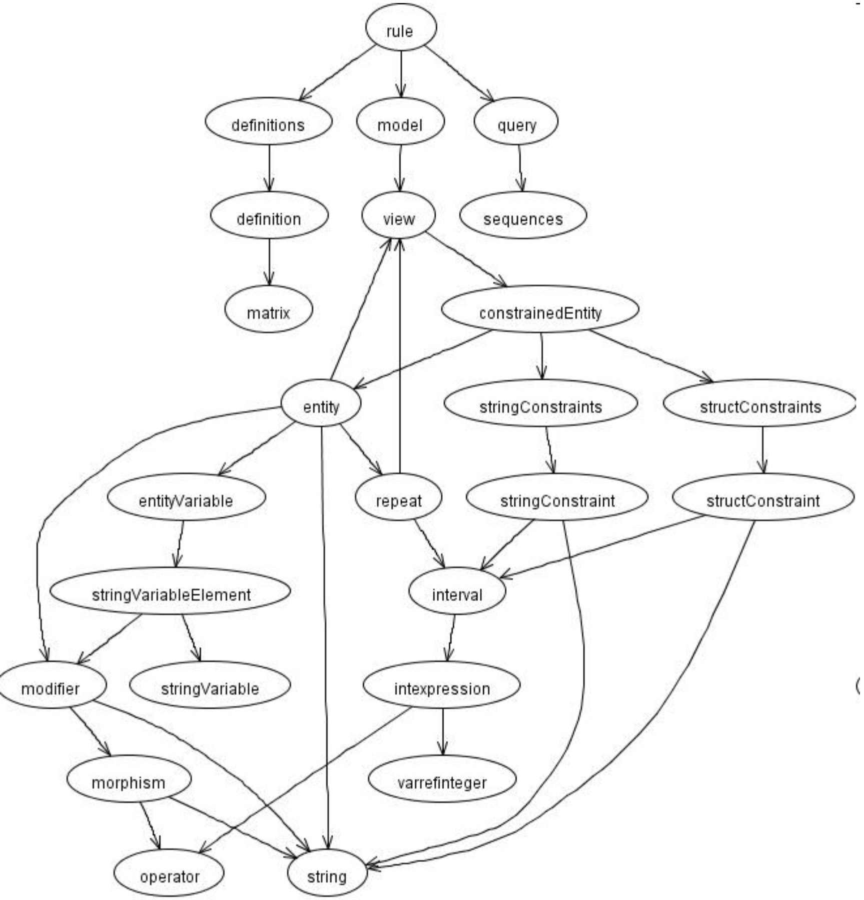


Functional examples

- I want to find X with a size [10,20], 3 codons, then X with a cost of 2 codons
  - X { #[10,20]} , Y {#[3,3]} , ?X { $2 }
- I want to find X repeated 2 or 3 times, with a possible cost of 1
  - repeat(?X { $1})+[2,3]

Examples above are functional examples, grammar is not Logol compliant

Grammar description



Main features:

- save and reuse a partial match
- morphisms (word complement and inverse)
- no left to right constraints (or so few…)
- parental constraints
    - Ex: X and Y have same parent with different cost constraints
- string constraints (start, end, size, content)
- structure constraints (cost, distance)
- notions of model with parameters
- repetitions
- overlapping between words
- operations on variables
    - Ex: starts at X position + 100
- Apply multiple models on same sequence to validate a match

Exemple:

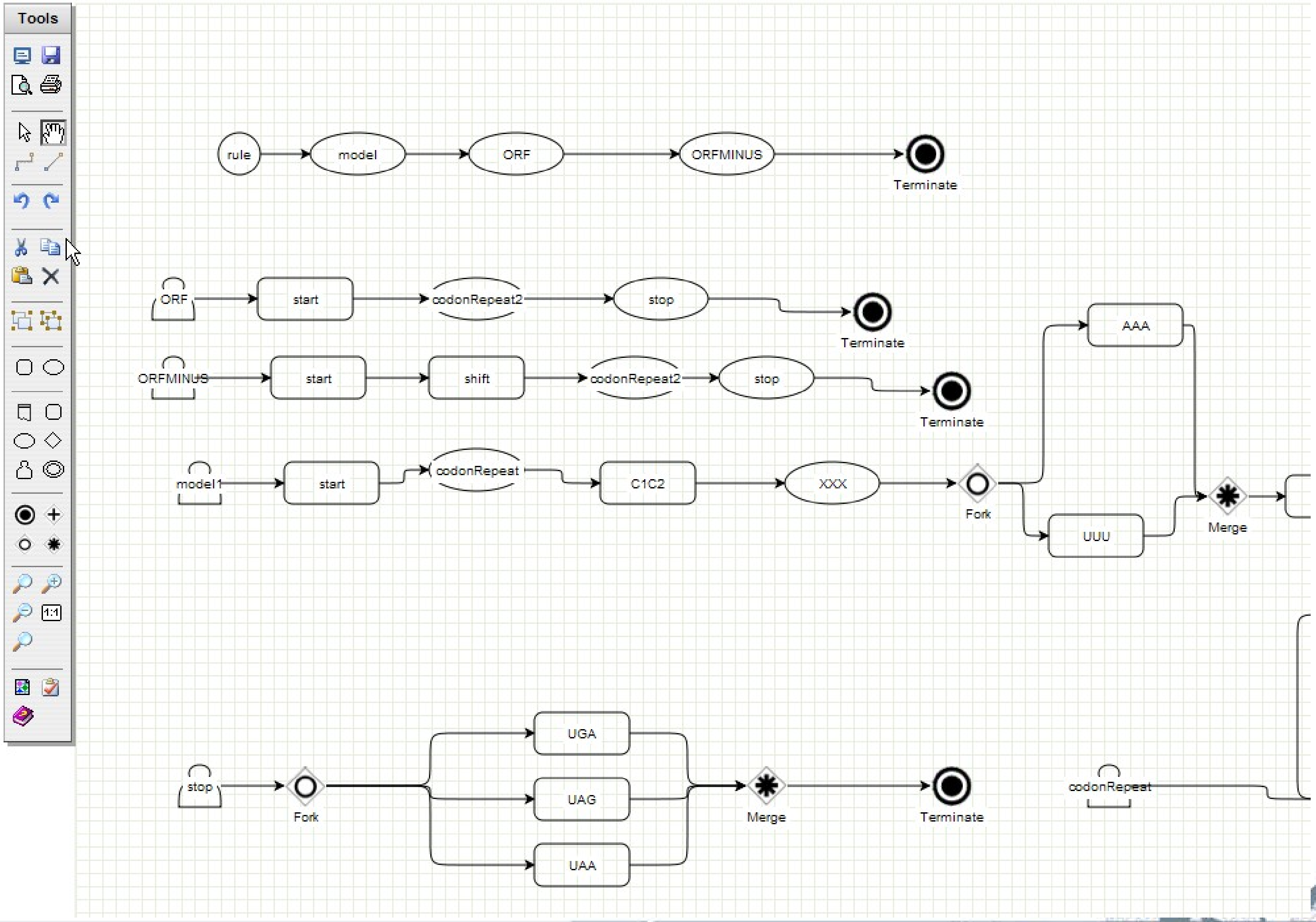
```
mod1(X)==>"aug":{_X}, ),?X,(("aaa")|("uuu")),! "g":{#[1,1]},
```

Tools: LogolDesigner

- It is a graphical interface to build in a web browser the Logol grammar. Logol grammar is not easy to learn, and can be complex to read for large models due to the tools writing constraints.
- Designer provides visual editing with printing and zoom capabilities.

Logol Designer

Web Model Designer



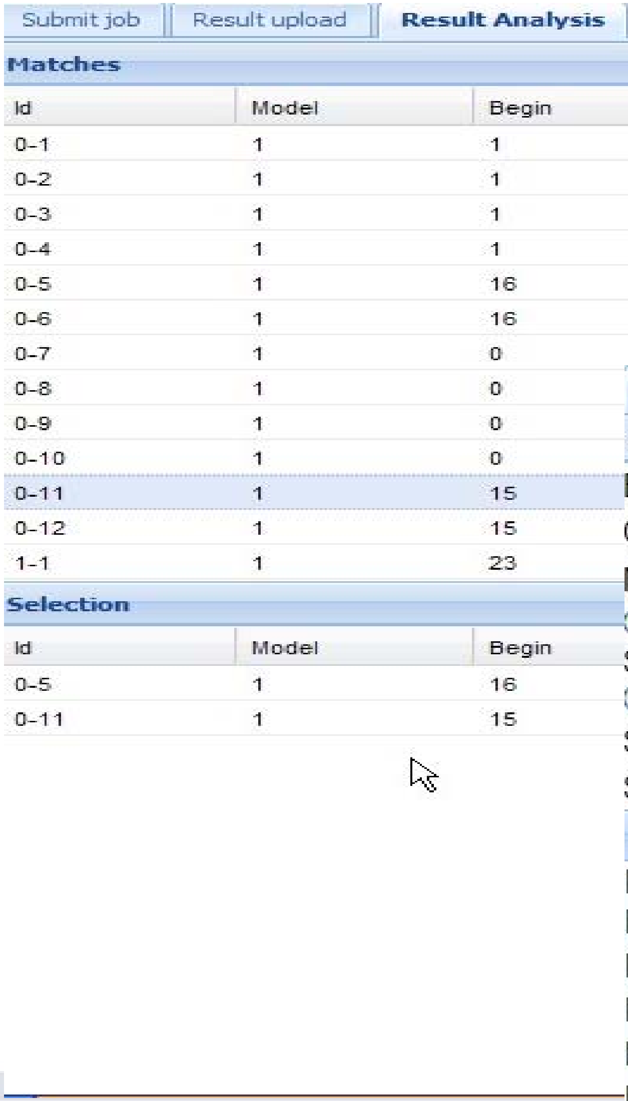
Tools: LogolMatch

- Core of the search
  - Takes as input a Logol graphical model or a Logol grammar file
  - Runs on a computer or a grid (linux)
    - Configurable to support multi-core architectures and to use multiple nodes to parallelize treatments when possible
    - Fast search, works on large sequences
      - Duration depends on pattern AND sequence (number of combinations, number and position of anchors), could run seconds or days
  - Outputs match occurrences in a compressed XML file
    - Each match is described with the details of the match (position of each word, size, number of errors compared to model…)
    - Possibility to convert it to Fasta (sequence only) or GFF output

Tools: LogolAnalyser

Online job submission(grid) and result analysis:

- Matches selection
- Fasta or GFF conversion
- Match display in a tree

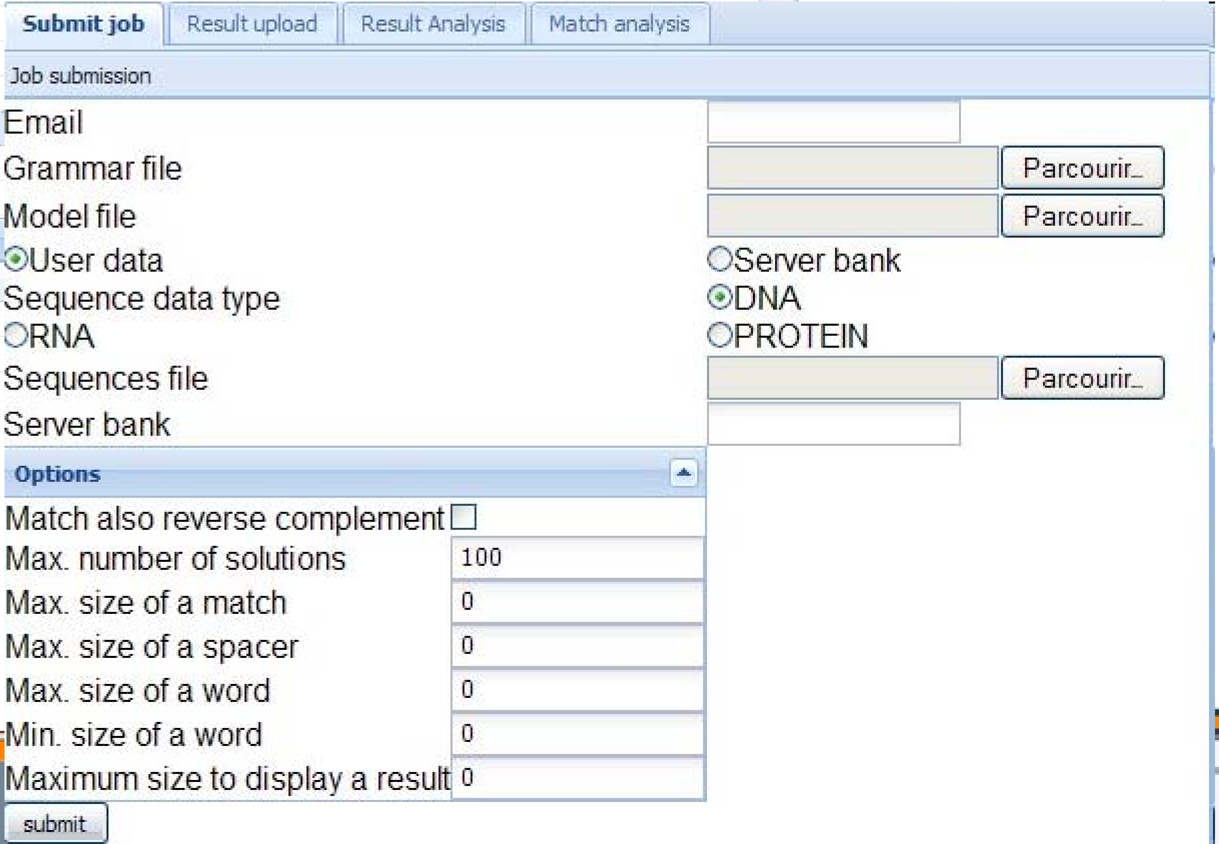


Matches

Id	Model	Begin
0-1	1	1
0-2	1	1
0-3	1	1
0-4	1	1
0-5	1	16
0-6	1	16
0-7	1	0
0-8	1	0
0-9	1	0
0-10	1	0
0-11	1	15
0-12	1	15
1-1	1	23

Selection

Id	Model	Begin
0-5	1	16
0-11	1	15



Job submission

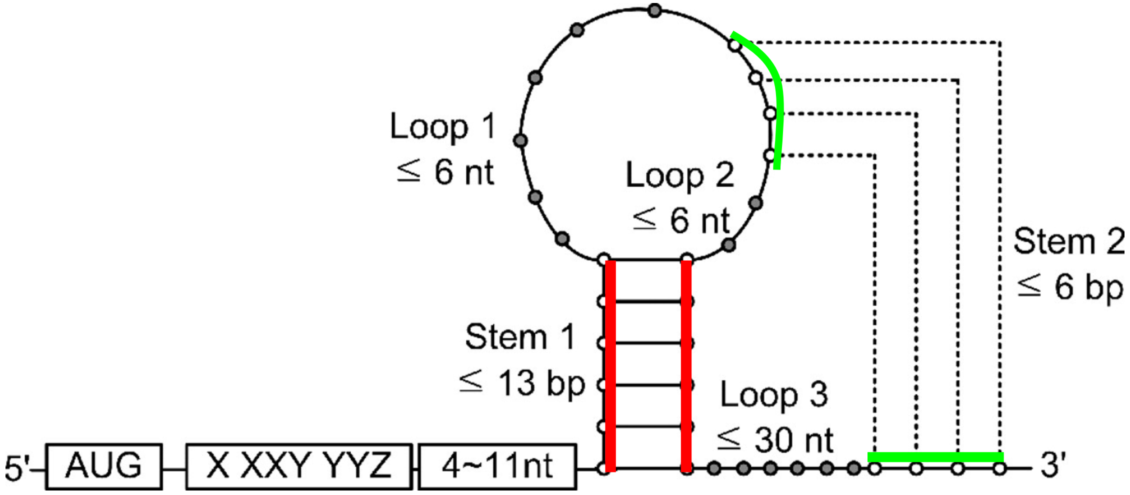
Email	
Grammar file	
Model file	
● User data	○ Server bank
Sequence data type	● DNA
○ RNA	○ PROTEIN
Sequences file	
Server bank	

Options

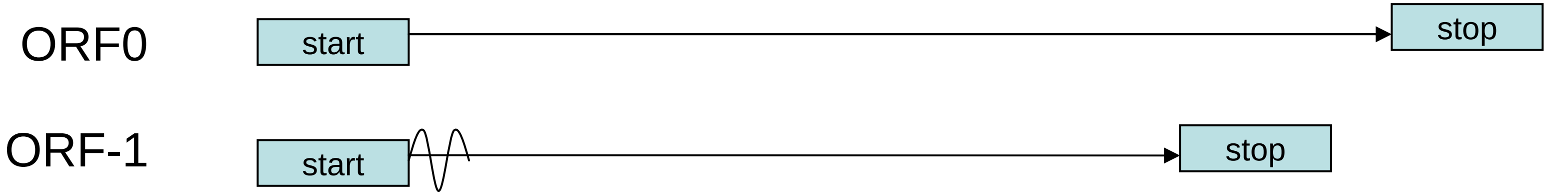
Match also reverse complement	□
Max. number of solutions	100
Max. size of a match	0
Max. size of a spacer	0
Max. size of a word	0
Min. size of a word	0
Maximum size to display a result	0

Use case example

programmed -1 ribosomal frameshift signals



start Slippery site



ORF0

start

stop

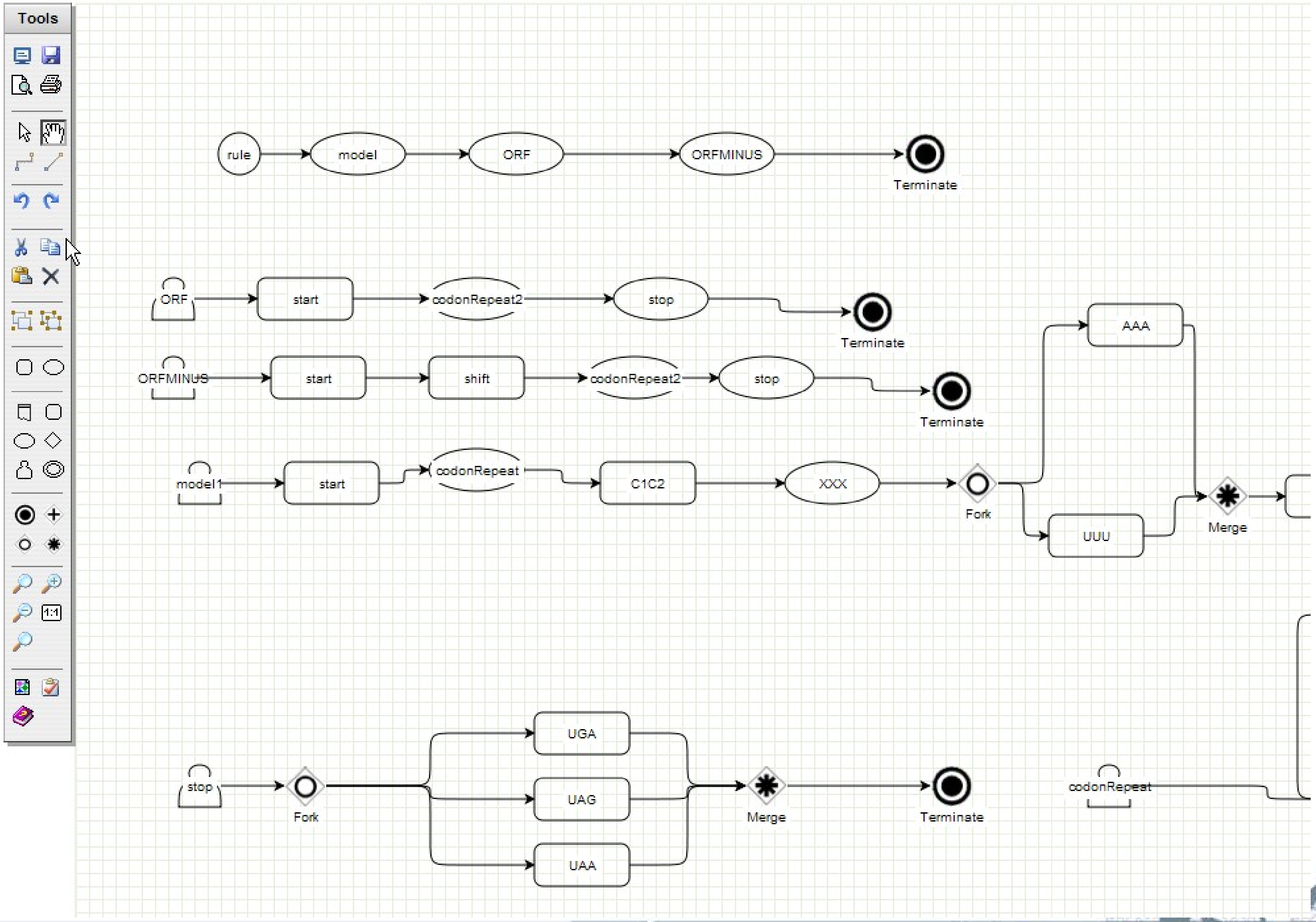
ORF-1

start

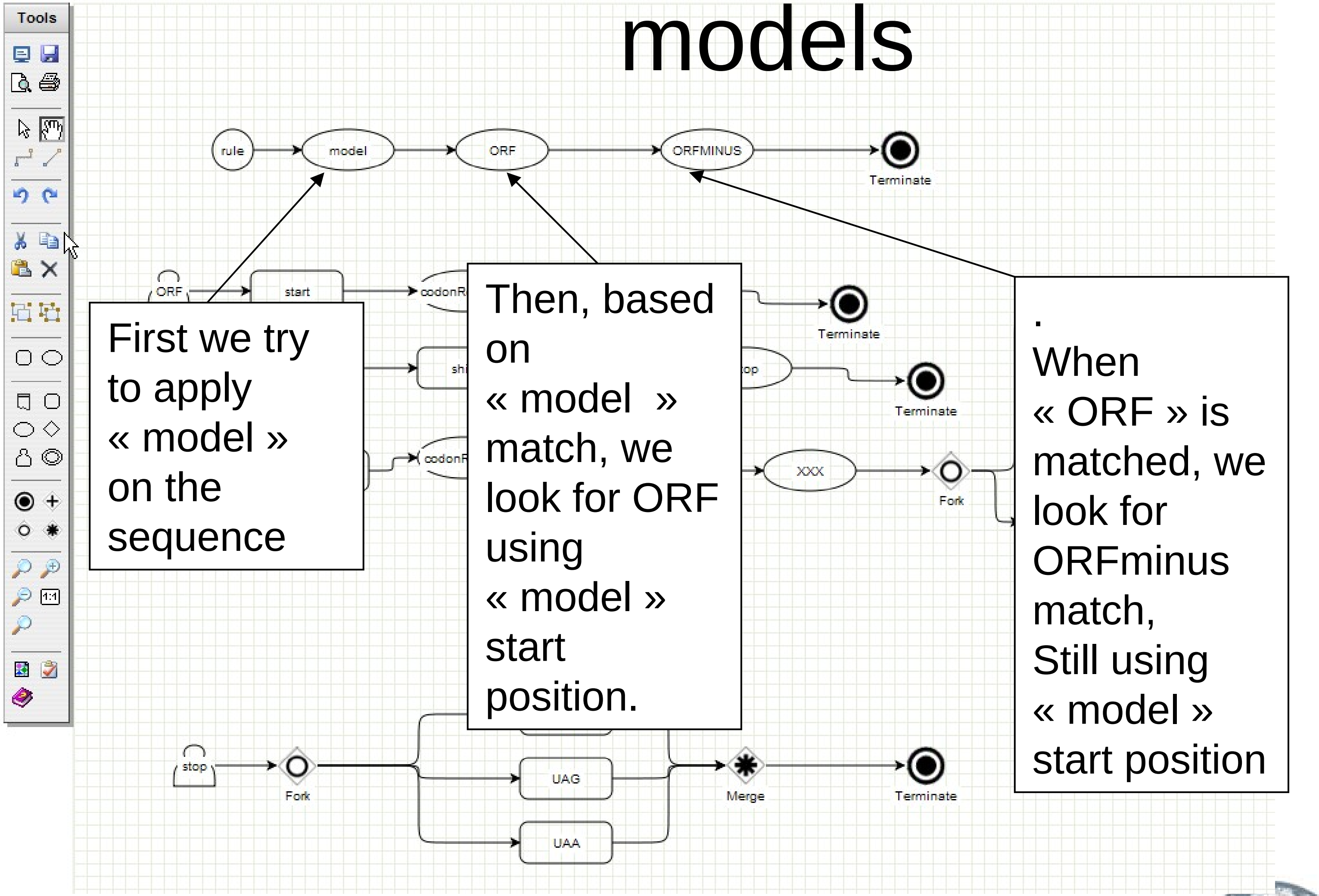
stop

2 codon slip -> phase change

Use case: Logol diagram



Use case: Search for several linked models

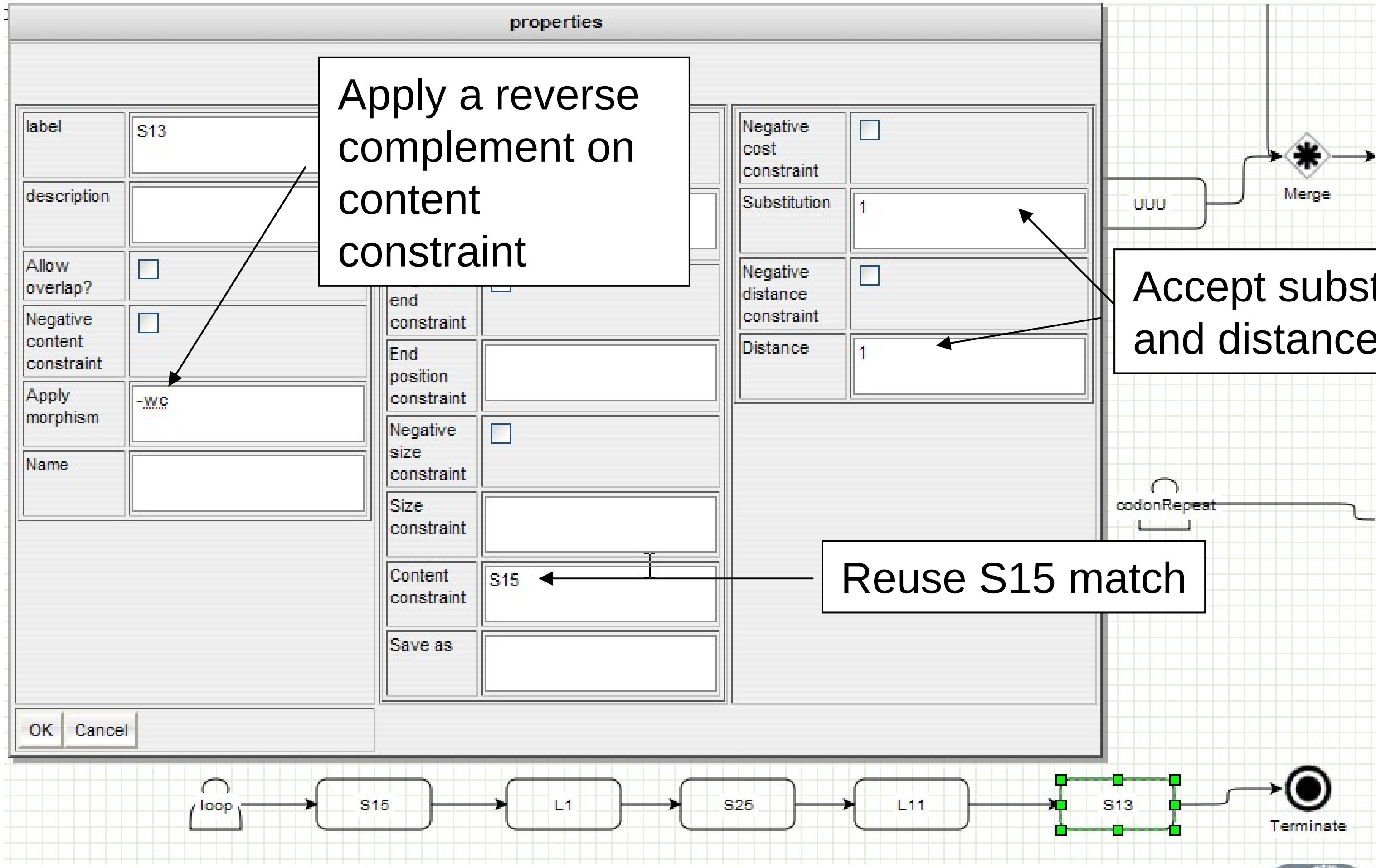


First we try to apply « model » on the sequence

Then, based on « model » match, we look for ORF using « model » start position.

.
When « ORF » is matched, we look for ORFminus match,
Still using « model » start position

Use case: Component properties



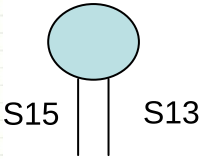
properties

label	S13
description	
Allow overlap?	□
Negative content constraint	□
Apply morphism	-wc
Name	
end constraint	□
End position constraint	
Negative size constraint	□
Size constraint	
Content constraint	S15
Save as	
Negative cost constraint	□
Substitution	1
Negative distance constraint	□
Distance	1

Apply a reverse complement on content constraint

Accept substition and distance

Reuse S15 match



Use case: Grammar

Automatic conversion of previous model, this is the Logol grammar equivalent:

- mod1(LOGOLVAR24). mod8(LOGOLVAR24). mod10(LOGOLVAR24)==*>SEQ1

- mod1(LOGOLVAR1)==>"aug":{_LOGOLVAR1%start%},mod2%codonRepeat%(LOGOLVAR2),LOGOLVAR3%CC%:{#[2,2]},mod3%xxx%(LOGOLVAR2),(("aaa")|("uuu")),!"g":{#[1,1]},LOGOLVAR4%spacer%:{#[3,9]},mod4%loop%(LOGOLVAR5%S25%),LOGOLVAR6%L2%:{#[0,100]},-"wc" ?LOGOLVAR5%S25%:{$[0,1],£[0,1]},mod5%spacerstop%(LOGOLVAR2)
- mod2(LOGOLVAR7)==>repeat(mod6%notstop%(LOGOLVAR7),[0,0])+[0,300]
- mod7(LOGOLVAR8)==>(("uga")|("uag")|("uaa"))
- mod4(LOGOLVAR9)==>LOGOLVAR10%S%:{#[6,100],_LOGOLVAR11%S15%},LOGOLVAR12%L1%:{#[0,2]},LOGOLVAR13%S2%:{#[5,100],_LOGOLVAR9%S25%},LOGOLVAR14%L11%:{#[0,2]},-"wc" ?LOGOLVAR11%S15%:{$[0,1],£[0,1]}
- mod6(LOGOLVAR15)==>!(mod7%stop%(LOGOLVAR15)):{#[3,3]}
- mod3(LOGOLVAR16)==>(("aaa")|("ccc")|("uuu")|("ggg"))
- mod5(LOGOLVAR17)==>((.*:{#[0,1000]},"uga")|(.*:{#[0,1000]},"uag")|(.*:{#[0,1000]},"uaa"))
- mod8(LOGOLVAR18)==>"aug":{@[@LOGOLVAR18%start%,@LOGOLVAR18%start%]},mod9%codonRepeat2%(LOGOLVAR19),mod7%stop%(LOGOLVAR19)
- mod10(LOGOLVAR20)==>"aug":{@[@LOGOLVAR20%start%,@LOGOLVAR20%start%]},LOGOLVAR21%shift%:{#[2,2]},mod9%codonRepeat2%(LOGOLVAR22),mod7%stop%(LOGOLVAR22)
- mod9(LOGOLVAR23)==>repeat(mod6%notstop%(LOGOLVAR23),[0,0])+[17,900]

Shortcoming features

- DNA ambiguity in pattern (not in sequence)
- Parameters auto-determination based on model
- Phase reference (A on same phase than B)
- Multiply and divide by operators
  - Example: #[#X / 2, #X] size constraint is [size of X/2, size of X]
- LogolDesigner: remote server model saving/reload (identification required)

- Other ideas are welcome !

Thank you!

Software is available on GenOUEST platform online:

http://webapps.genouest.org/LogolDesigner

Or on genocluster2 (account required):

```
. /local/env/envlogol.sh;./LogolMultiExec.sh
```