Task scheduling for dual-arm industrial robots through constraint programming

MiniZinc modeling and solver comparison

Tommy Kvant

Institute of Computer Science
Lund University

February 23, 2015

Outline

Introduction - YuMi®

- Dual-armed robot
- Flexible - multiple tools
- Fine motor skills

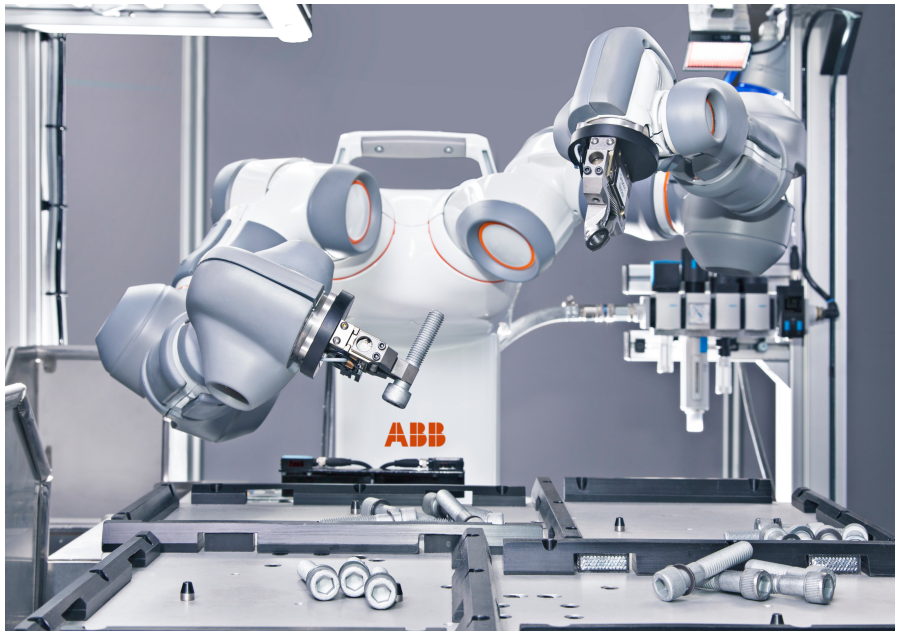


Photo: ABB

Introduction - Project goal

- Constraint Programming model for dual-armed robots such as YuMi®
  - Change tools
  - Carry only one component at the time
  - Same duration for tool changes, regardless of direction
  - Use trays, fixtures and outputs
- Implement the model in MiniZinc
- Test the model with 6 solvers and compare the results

Introduction - MiniZinc

- Declarative language
- Medium level
- Translates to FlatZinc
- Aims to be standard
- Many solvers can read FlatZinc

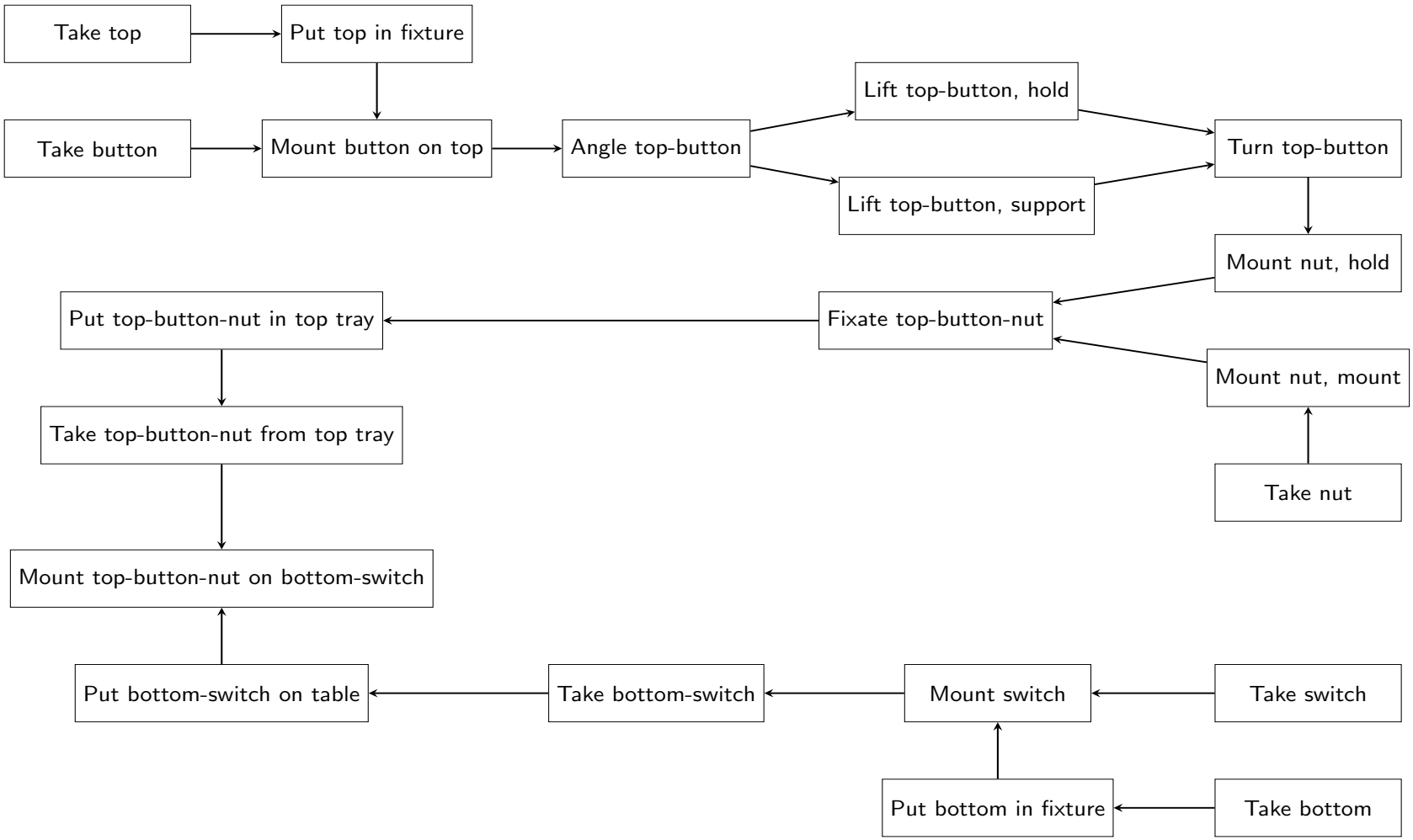
Case Study

Case Study

Physical Entities

- Machines
- Tools
- Components
- Tray
- Fixture
- Output

Assembly


Take top
Put top in fixture
Take button
Mount button on top
Angle top-button
Lift top-button, hold
Lift top-button, support
Turn top-button
Mount nut, hold
Fixate top-button-nut
Put top-button-nut in top tray
Mount nut, mount
Take top-button-nut from top tray
Take nut
Mount top-button-nut on bottom-switch
Put bottom-switch on table
Take bottom-switch
Mount switch
Take switch
Put bottom in fixture
Take bottom

Model

Job Shop Problem

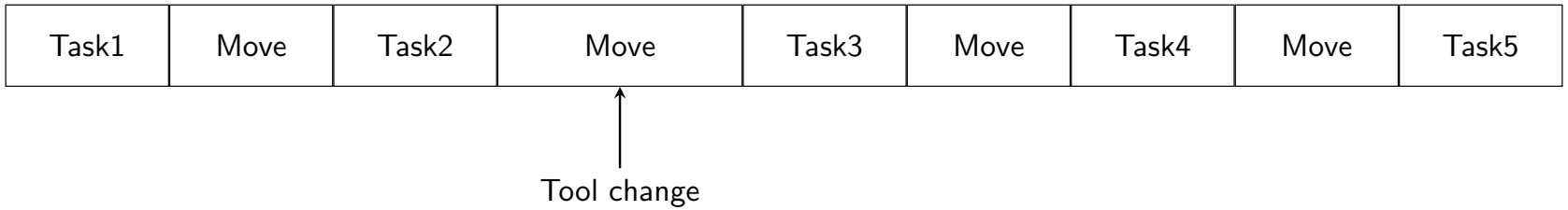
- n jobs, varying size
- m identical machines
- NP-complete for $m \geq 2$ and $n \geq 3$

Model

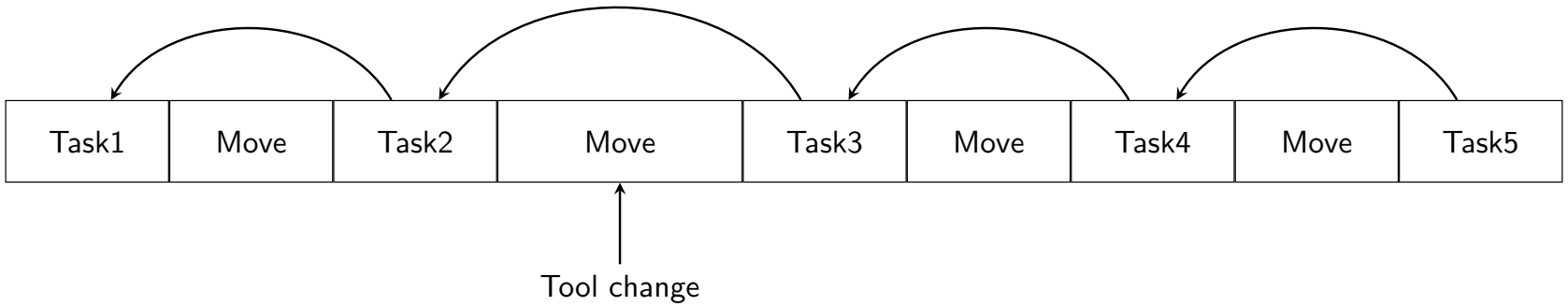
Model

Task1	Move	Task2	Move	Task3	Move	Task4	Move	Task5

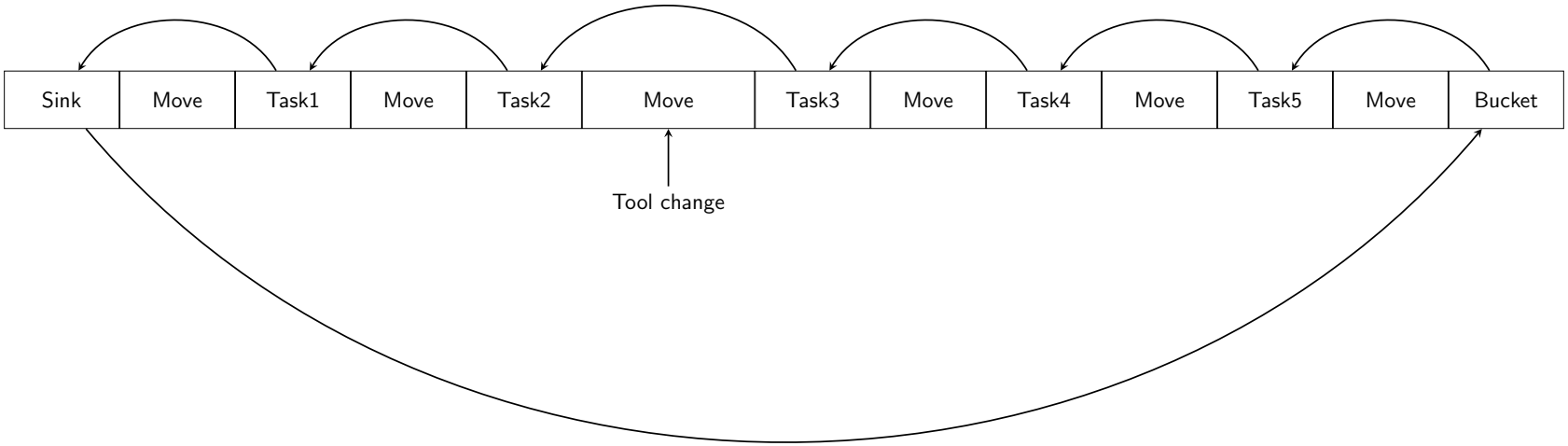
Model


Task1
Move
Task2
Move
Task3
Move
Task4
Move
Task5
Tool change

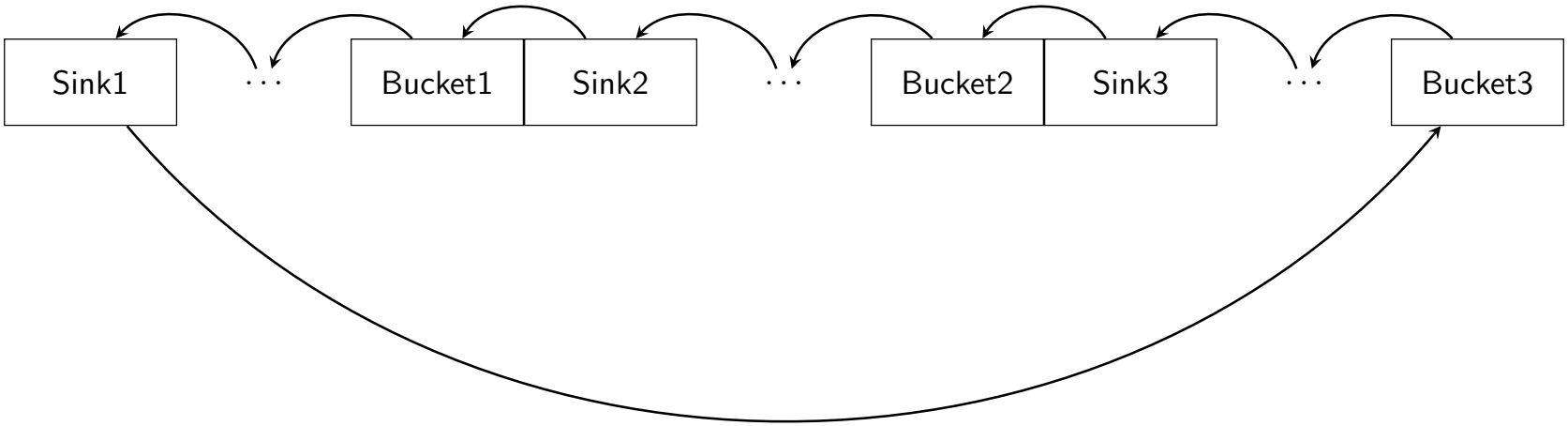
Model


Task1
Move
Task2
Move
Task3
Move
Task4
Move
Task5
Tool change

Model


Sink
Move
Task1
Move
Task2
Move
Task3
Move
Task4
Move
Task5
Move
Bucket
Tool change

Model



Components

Components

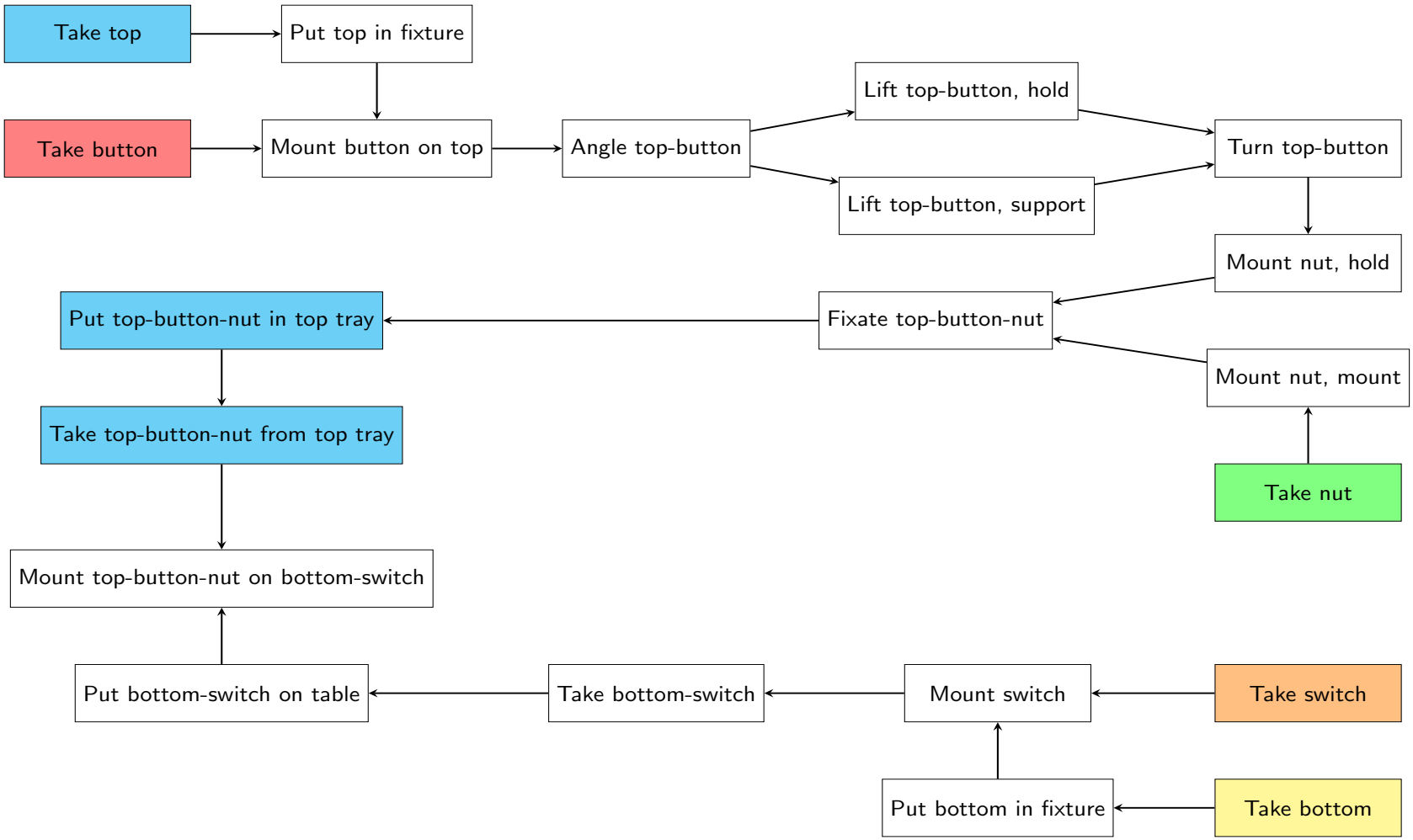
- Primitive components
- Sub-assemblies

Storage Mediums

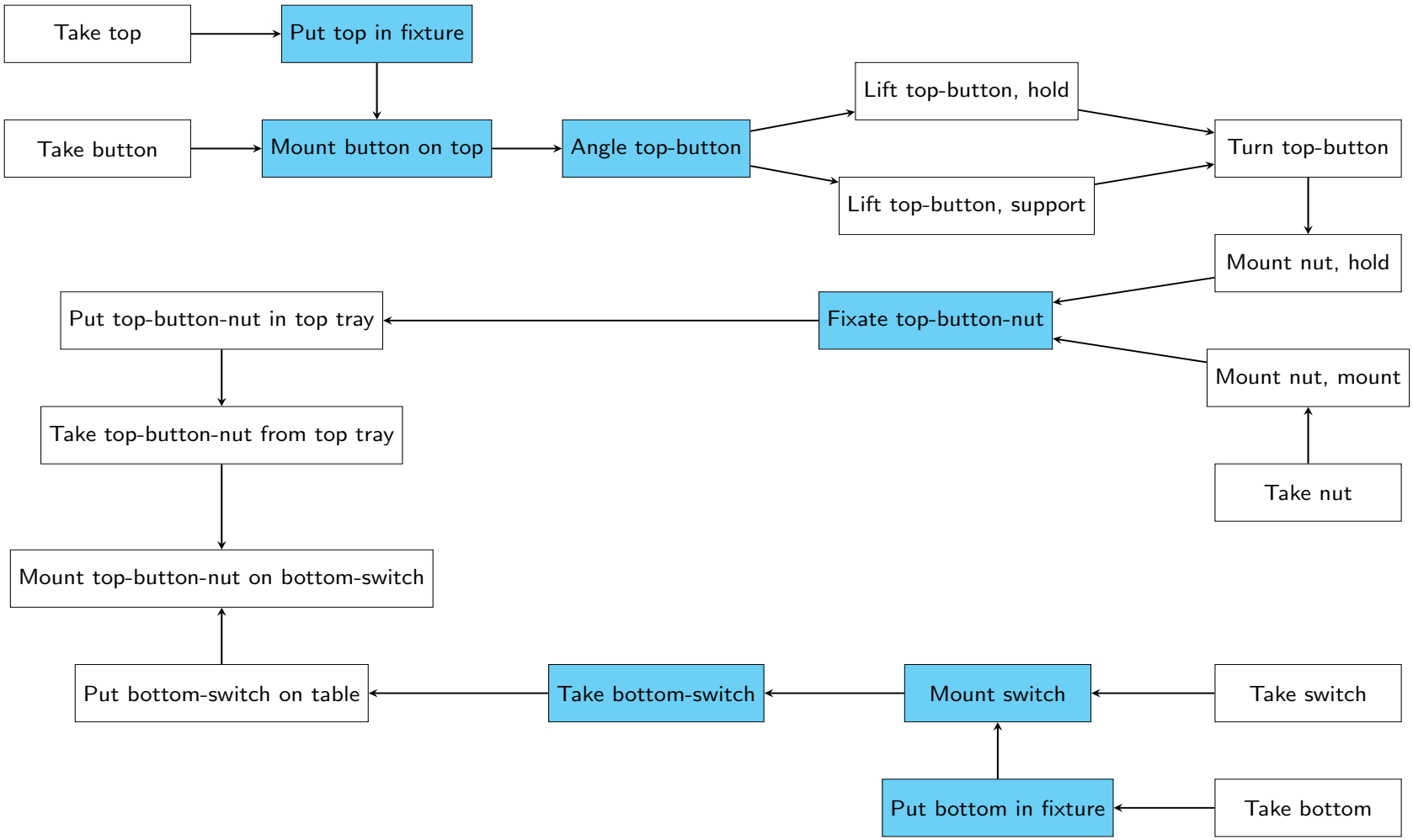
Storage mediums

- Tray - Top tray, Button tray, etc.
- Fixture
- Output

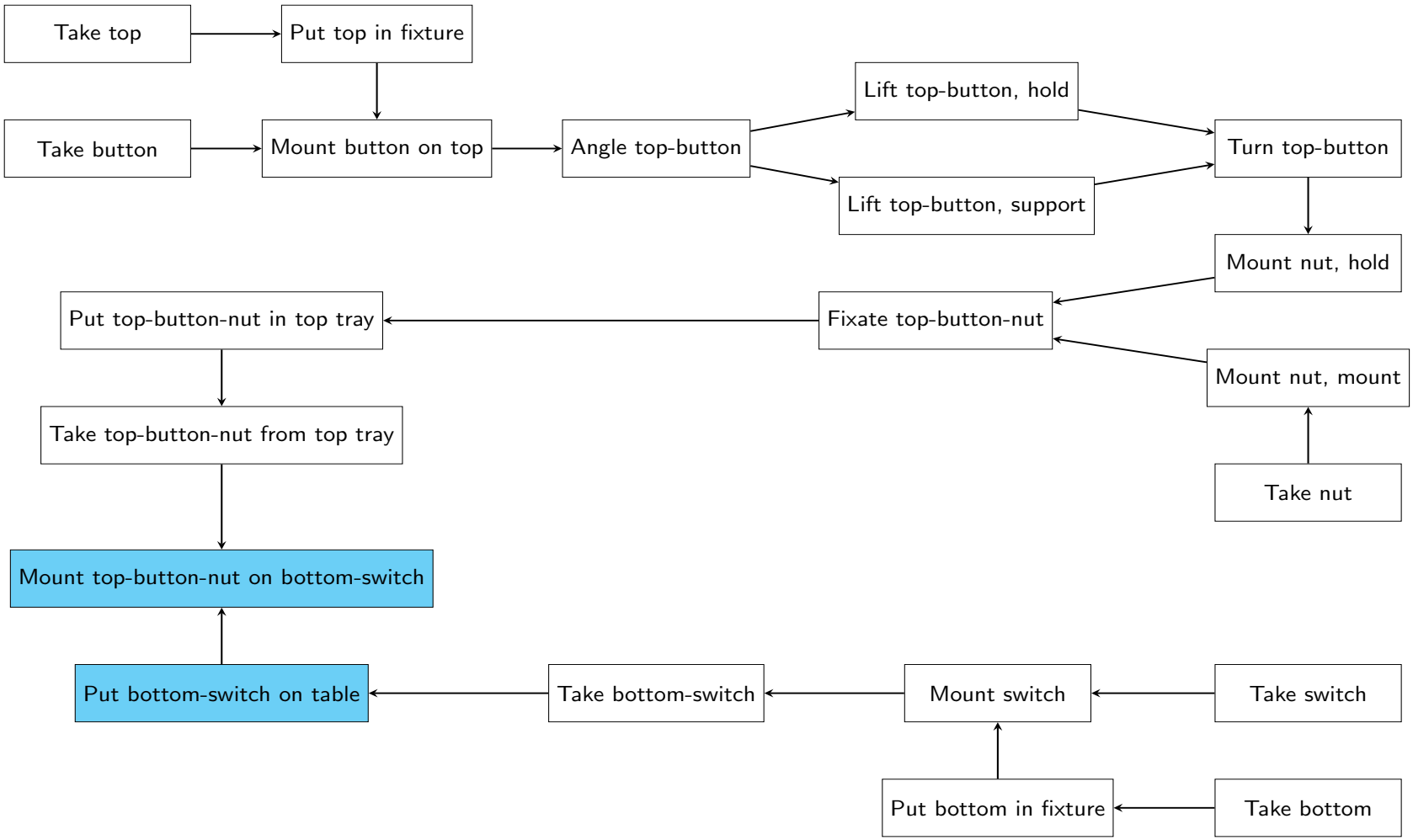
Tray


Take top
Put top in fixture
Take button
Mount button on top
Angle top-button
Lift top-button, hold
Lift top-button, support
Turn top-button
Mount nut, hold
Fixate top-button-nut
Put top-button-nut in top tray
Mount nut, mount
Take top-button-nut from top tray
Take nut
Mount top-button-nut on bottom-switch
Put bottom-switch on table
Take bottom-switch
Mount switch
Take switch
Put bottom in fixture
Take bottom

Fixture


Take top
Put top in fixture
Take button
Mount button on top
Angle top-button
Lift top-button, hold
Lift top-button, support
Turn top-button
Mount nut, hold
Fixate top-button-nut
Put top-button-nut in top tray
Mount nut, mount
Take top-button-nut from top tray
Take nut
Mount top-button-nut on bottom-switch
Put bottom-switch on table
Take bottom-switch
Mount switch
Take switch
Put bottom in fixture
Take bottom

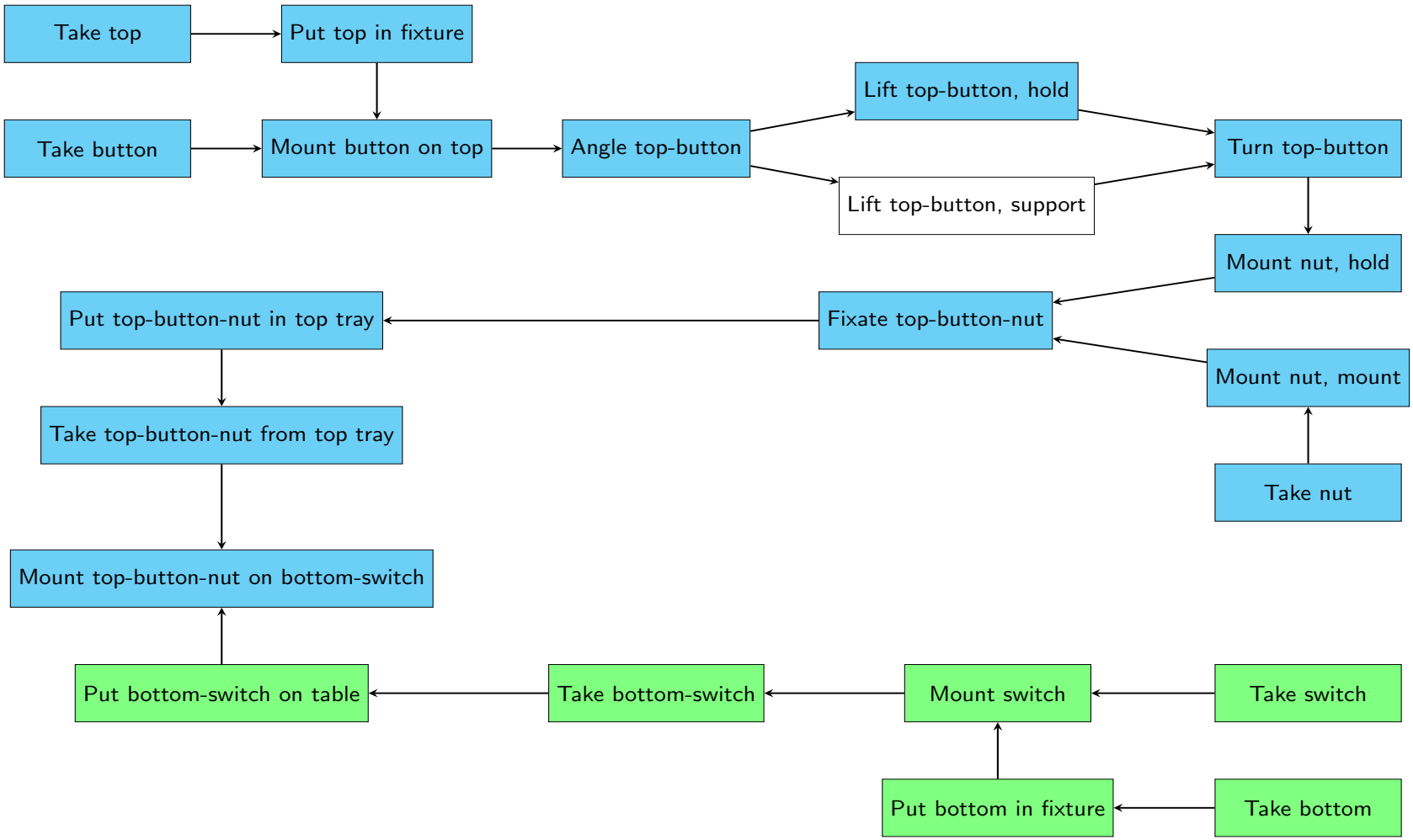
Output


Take top
Put top in fixture
Take button
Mount button on top
Angle top-button
Lift top-button, hold
Lift top-button, support
Turn top-button
Mount nut, hold
Fixate top-button-nut
Put top-button-nut in top tray
Mount nut, mount
Take top-button-nut from top tray
Take nut
Mount top-button-nut on bottom-switch
Put bottom-switch on table
Take bottom-switch
Mount switch
Take switch
Put bottom in fixture
Take bottom

Tools

- Tools available
- Tool used by tasks

Tools

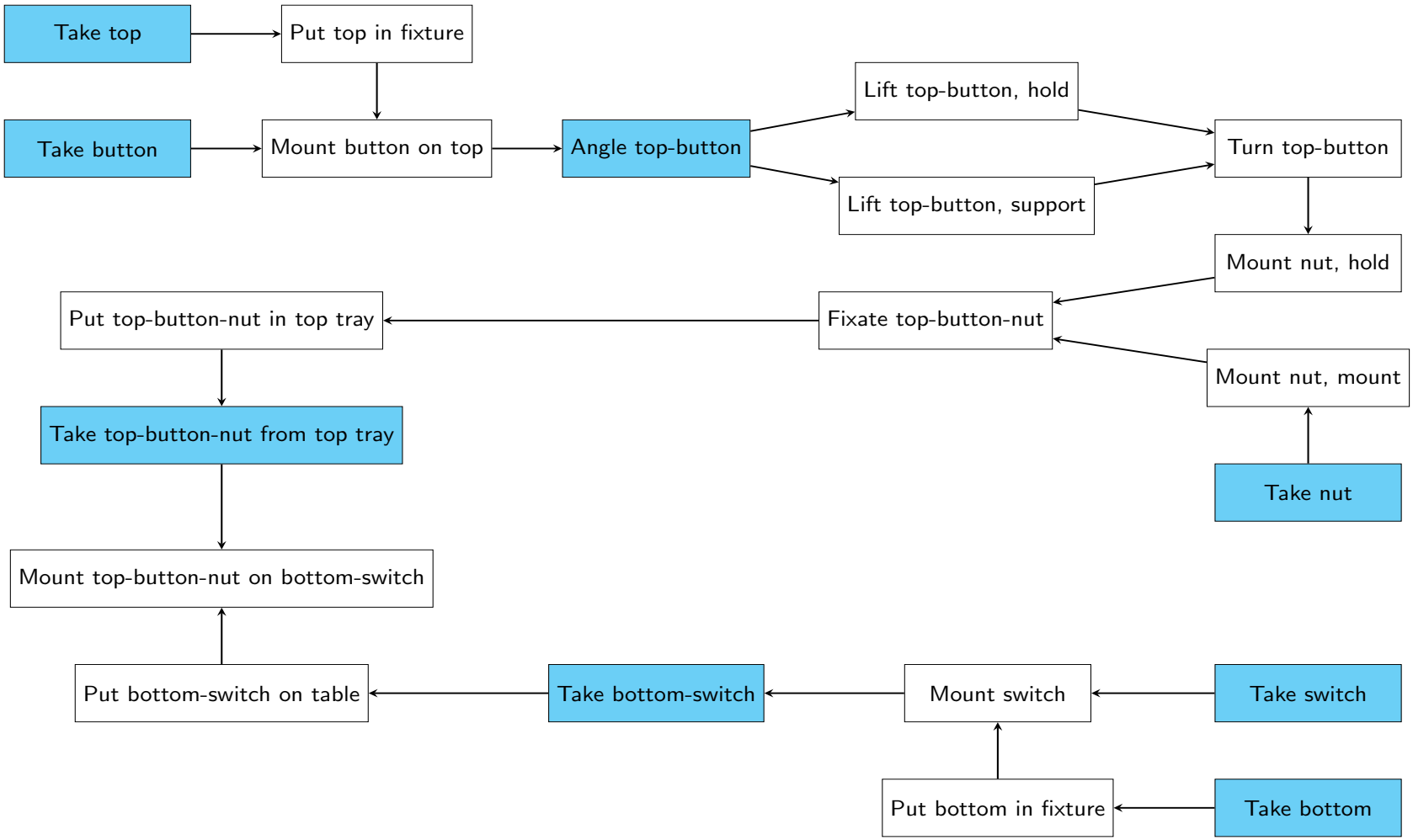

Take top
Put top in fixture
Take button
Mount button on top
Angle top-button
Lift top-button, hold
Lift top-button, support
Turn top-button
Mount nut, hold
Fixate top-button-nut
Put top-button-nut in top tray
Mount nut, mount
Take top-button-nut from top tray
Take nut
Mount top-button-nut on bottom-switch
Put bottom-switch on table
Take bottom-switch
Mount switch
Take switch
Put bottom in fixture
Take bottom

Action

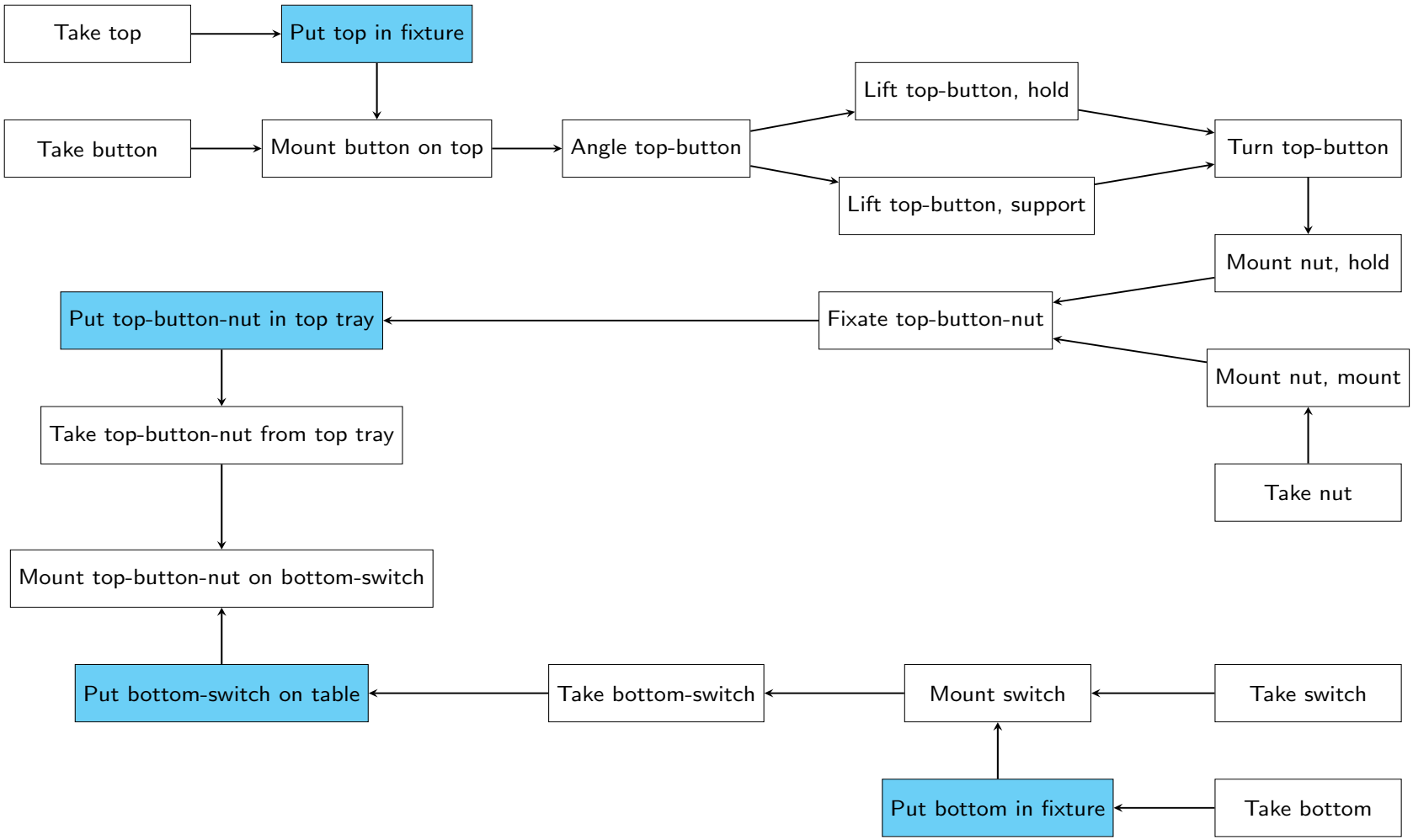
Task actions

- Taking
- Mounting
- Putting
- Moving

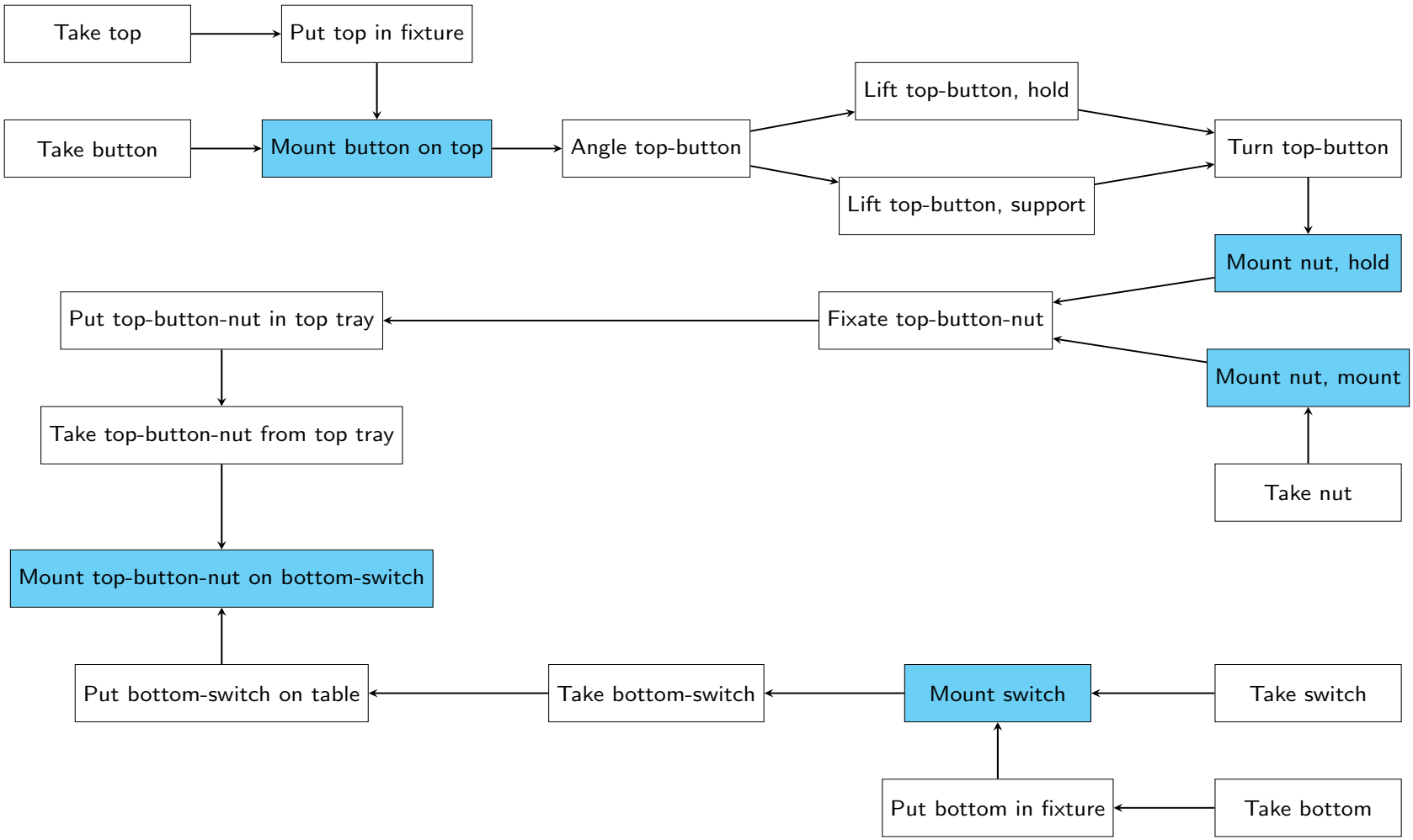
Taking


Take top
Put top in fixture
Take button
Mount button on top
Angle top-button
Lift top-button, hold
Lift top-button, support
Turn top-button
Mount nut, hold
Fixate top-button-nut
Put top-button-nut in top tray
Mount nut, mount
Take top-button-nut from top tray
Take nut
Mount top-button-nut on bottom-switch
Put bottom-switch on table
Take bottom-switch
Mount switch
Take switch
Put bottom in fixture
Take bottom

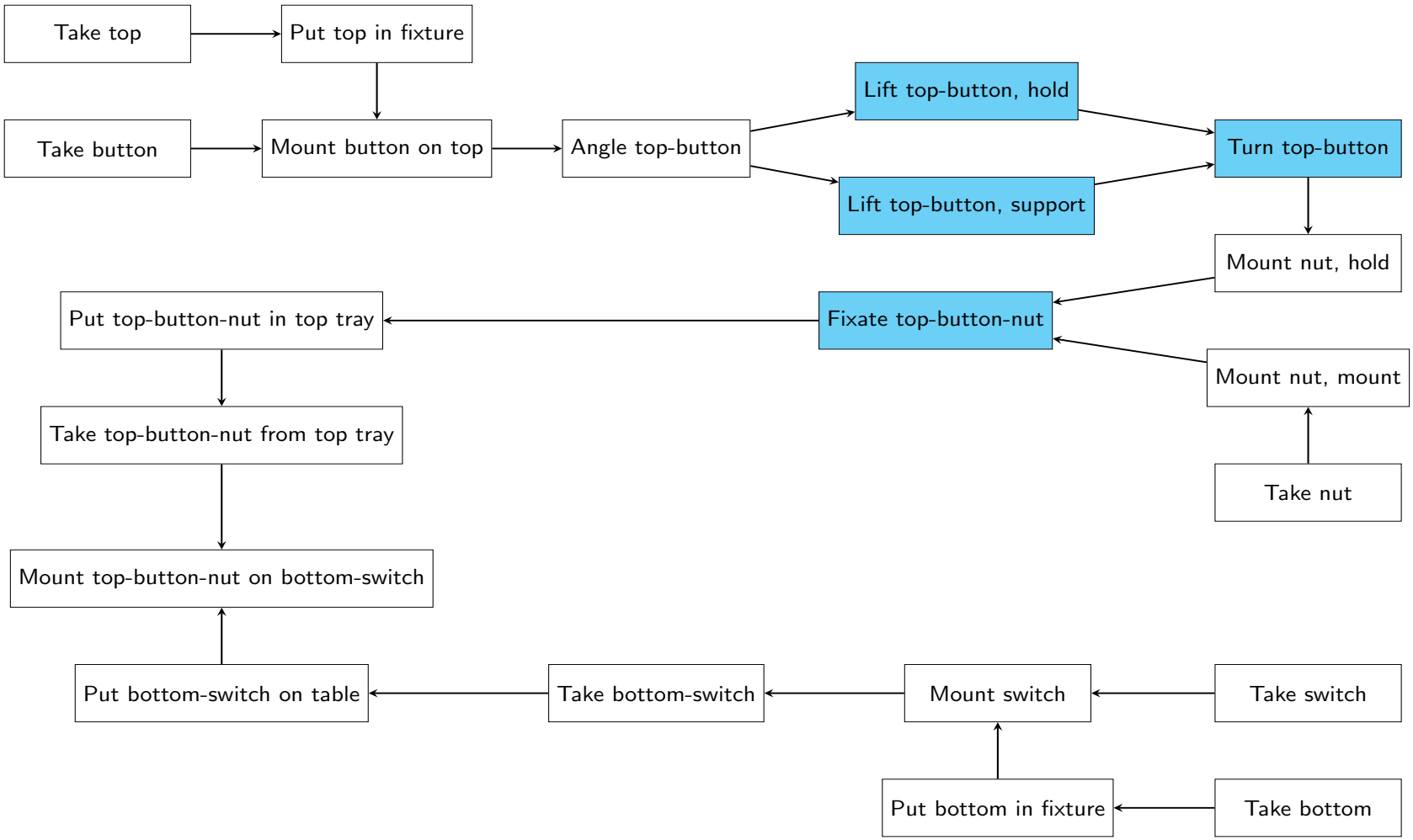
Putting


Take top
Put top in fixture
Take button
Mount button on top
Angle top-button
Lift top-button, hold
Lift top-button, support
Turn top-button
Mount nut, hold
Fixate top-button-nut
Put top-button-nut in top tray
Mount nut, mount
Take top-button-nut from top tray
Take nut
Mount top-button-nut on bottom-switch
Put bottom-switch on table
Take bottom-switch
Mount switch
Take switch
Put bottom in fixture
Take bottom

Mounting


Take top
Put top in fixture
Take button
Mount button on top
Angle top-button
Lift top-button, hold
Lift top-button, support
Turn top-button
Mount nut, hold
Fixate top-button-nut
Put top-button-nut in top tray
Mount nut, mount
Take top-button-nut from top tray
Take nut
Mount top-button-nut on bottom-switch
Put bottom-switch on table
Take bottom-switch
Mount switch
Take switch
Put bottom in fixture
Take bottom

Moving

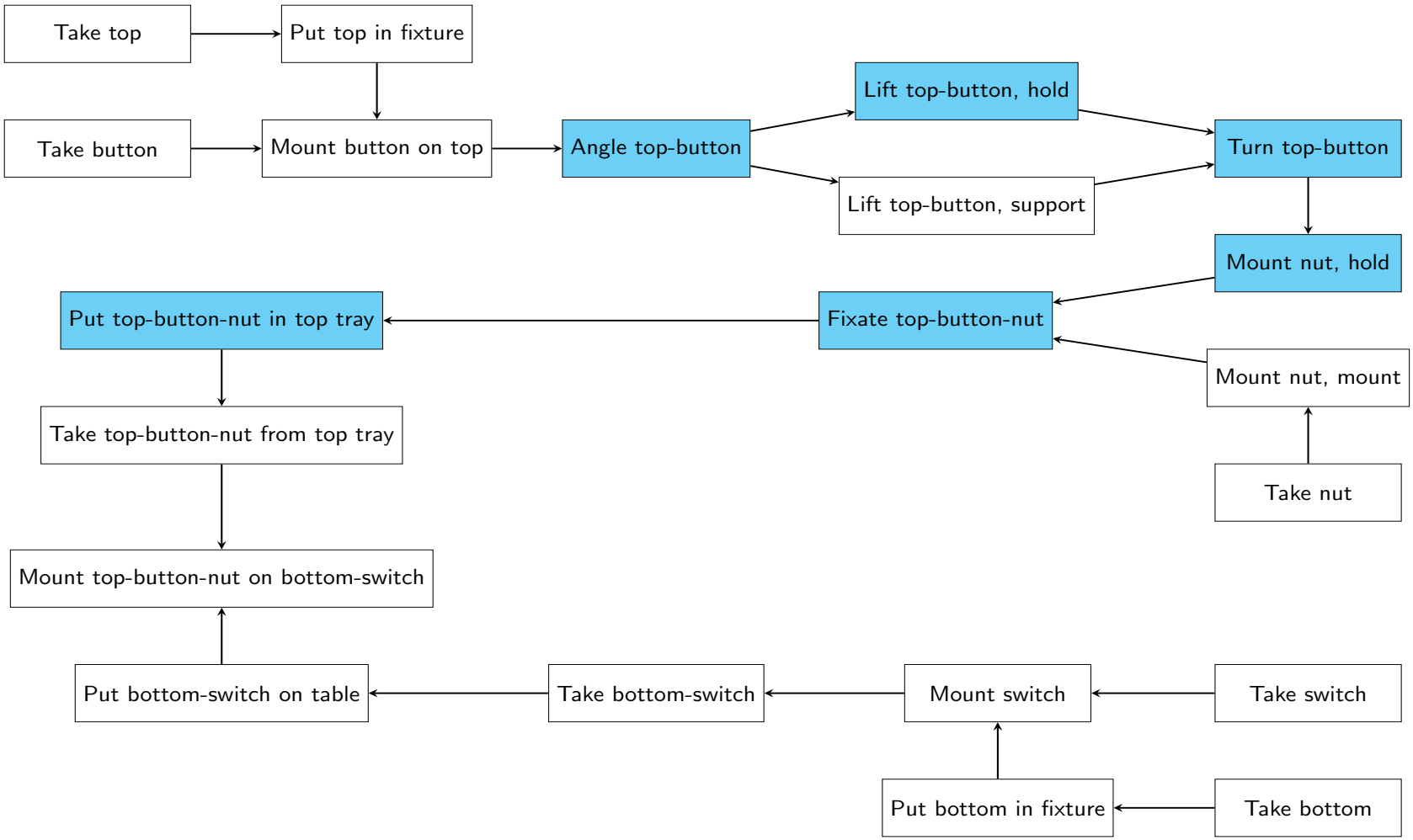

Take top
Put top in fixture
Take button
Mount button on top
Angle top-button
Lift top-button, hold
Lift top-button, support
Turn top-button
Mount nut, hold
Fixate top-button-nut
Put top-button-nut in top tray
Mount nut, mount
Take top-button-nut from top tray
Take nut
Mount top-button-nut on bottom-switch
Put bottom-switch on table
Take bottom-switch
Mount switch
Take switch
Put bottom in fixture
Take bottom

Model

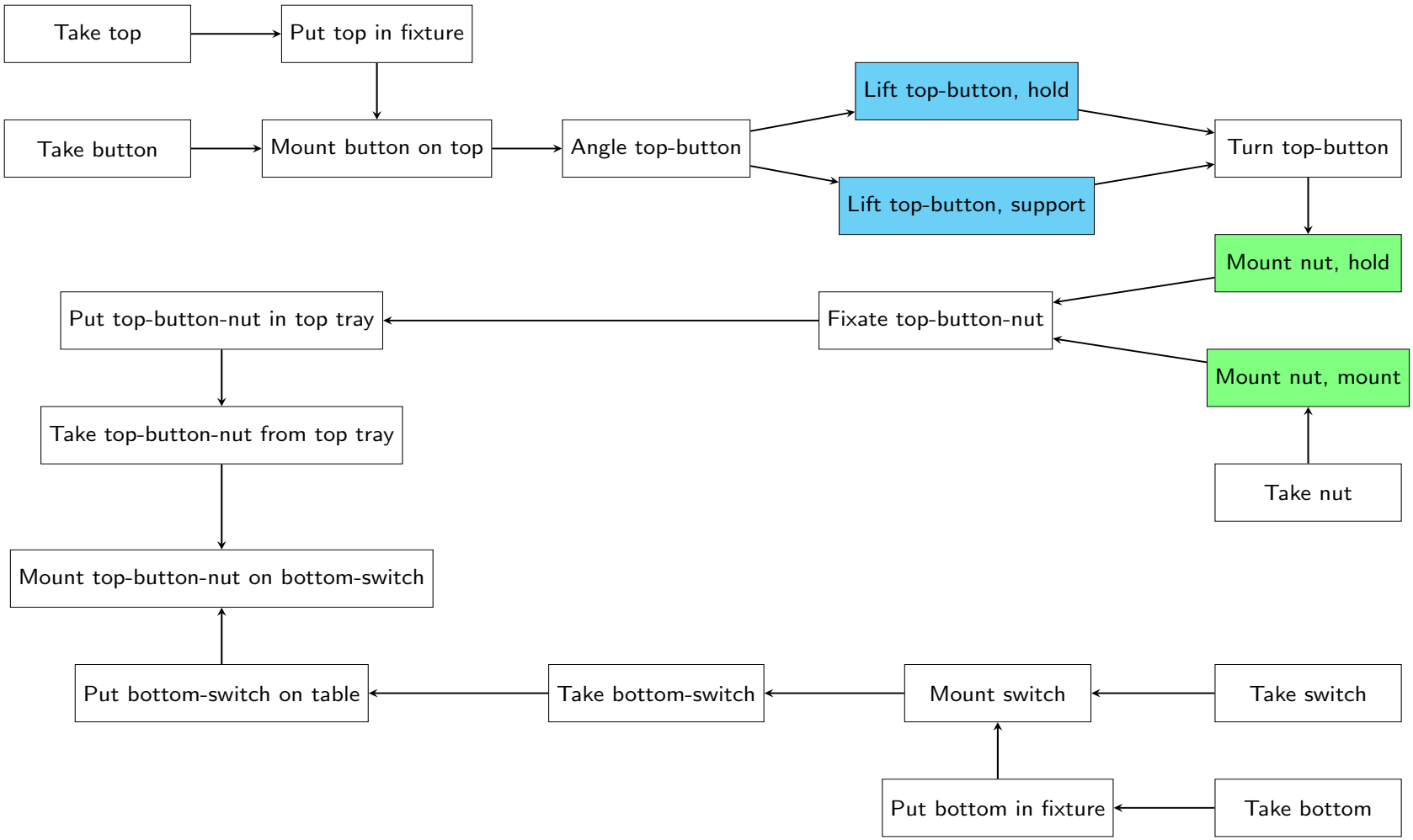
Group tasks

- Ordered group
- Concurrent group

Ordered group



Concurrent group



Filter

- Temporal filter
- Predecessor filter

Evaluation

- Test with 6 solvers
  - Solver time
  - FlatZinc file
- MiniZinc 1.6 & 2.0.1
- Combination of filters

Criteria

- FlatZinc parser
- Free

Solvers Tested

- G12/FD
- JaCoP
- Gecode
- or-tools
- Opturion CPX
- Choco3

G12/FD

- G12 Team, NICTA
- Mercury
- Default solver for MiniZinc

JaCoP

- Java Constraint Programming solver
- Open Source
- Developed since 2001 - Krzysztof Kuchcinski & Radoslaw Szymanek
- Silver medal

Gecode

- C++
- Open Source
- Christian Schulte
- Parallel searches - utilising multiple cores
- All gold medals 2008-2012

or-tools

- C++
- Google - Operational Research
- Open Source
- Utilising multiple cores
- Gold medals 2013-2014

Opturion CPX

- Opturion Pty Ltd
- Commercial
- SAT combo
- Gold medals 2013, all silver medals 2014

Choco3

- Java
- Open Source
- Developed since early 2000 - Jean-Guillaume Fages & Charles Prud'homme
- Not same as predecessor Choco2

Assembly Times

Manual Time

516 t.u.

Assembly Times

Manual Time	Solver Time
516 t.u.	512 t.u.

Solver Time

	Pred & Temp		Pred		Temp		None	
	1.6	2.0.1	1.6	2.0.1	1.6	2.0.1	1.6	2.0.1
G12/FD	-	-	-	-	-	-	-	-
JaCoP	658	-	1011156	-	-	-	-	-
Gecode	-	60	-	71761	-	99	-	71186
or-tools	271	!	380	!	302	!	457	!
Opturion CPX	-	!	-	!	-	!	-	!
Choco3	-	-	-	-	-	-	-	-

Solver Time

	Pred							
	1.6							
G12/FD	-	-	-	-	-	-	-	-
JaCoP	658	-	1011156	-	-	-	-	-
Gecode	-	60	-	71761	-	99	-	71186
or-tools	271	!	380	!	302	!	457	!
Opturion CPX	-	!	-	!	-	!	-	!
Choco3	-	-	-	-	-	-	-	-

Solver Time

	Pred							
	1.6							
G12/FD	-							
JaCoP	658	-	1011156	-	-	-	-	-
Gecode	-	60	-	71761	-	99	-	71186
or-tools	271	!	380	!	302	!	457	!
Opturion CPX	-	!	-	!	-	!	-	!
Choco3	-	-	-	-	-	-	-	-

Solver Time

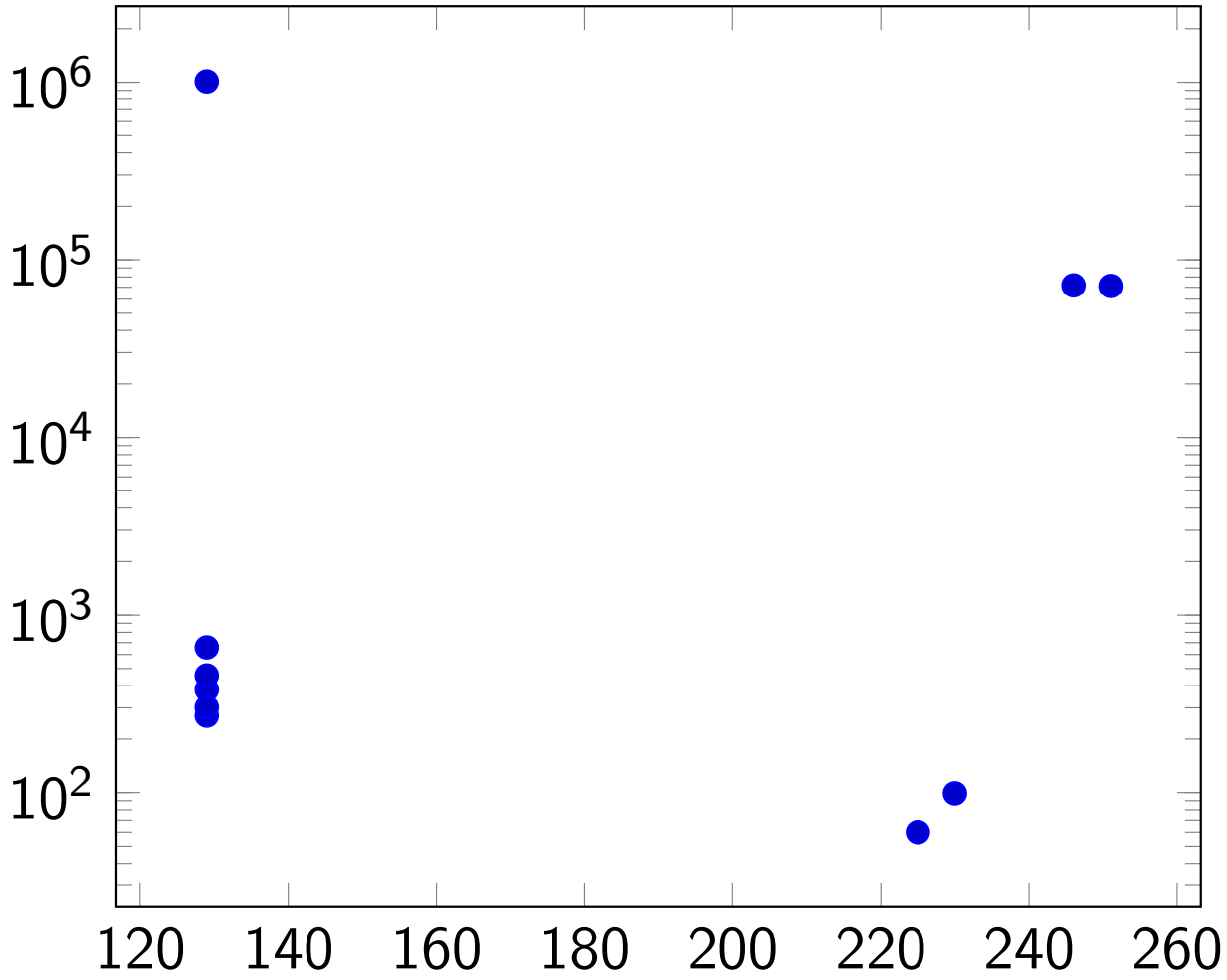
	Pred & Temp		Pr					
	1.6	2.0.1	1.6					
G12/FD	-	-	-					
JaCoP	658	-	1011156					
Gecode	-	60	-	71761	-	99	-	71186
or-tools	271	!	380	!	302	!	457	!
Opturion CPX	-	!	-	!	-	!	-	!
Choco3	-	-	-	-	-	-	-	-

Solver Time

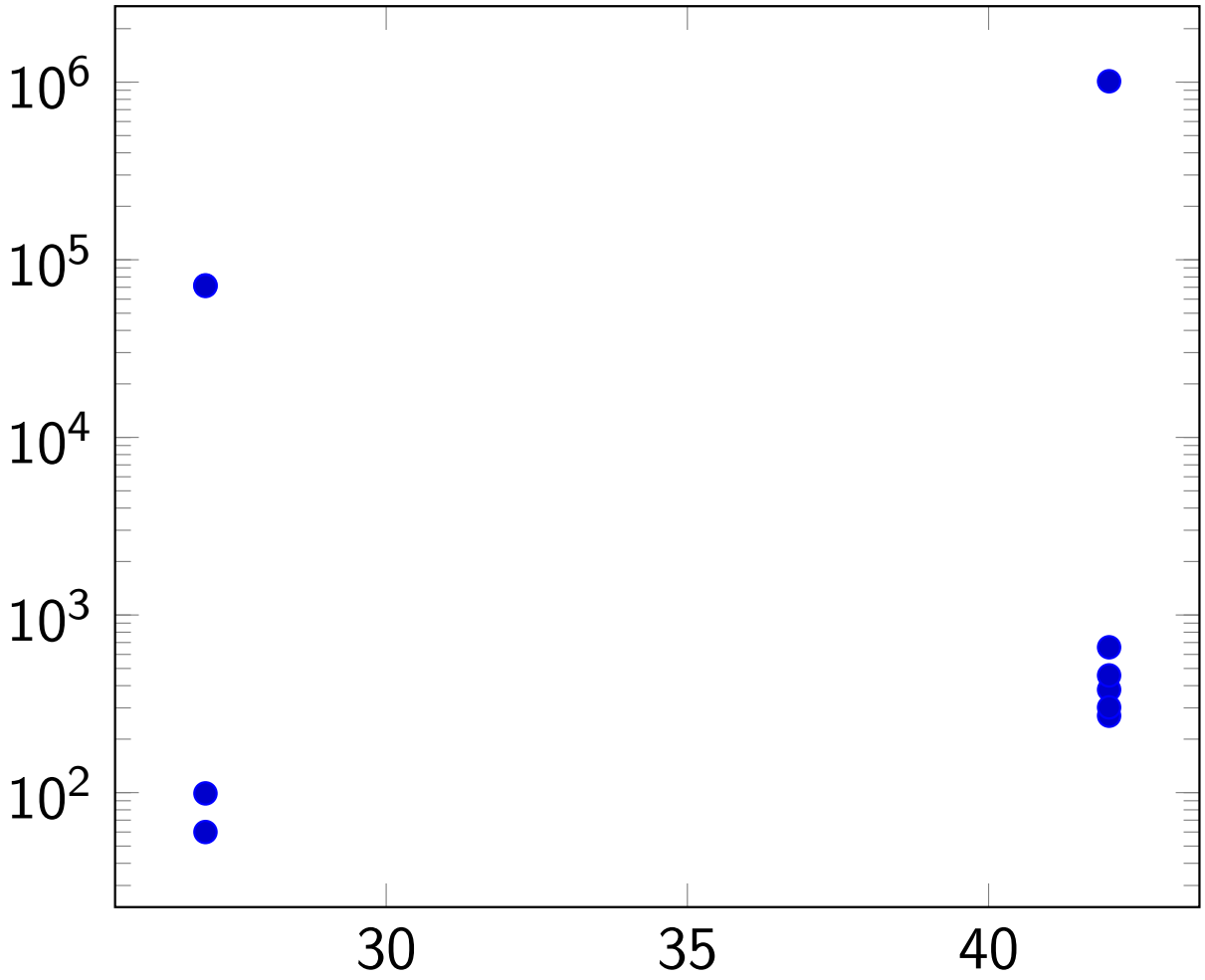
	Pred & Temp		Pr				
	1.6	2.0.1	1.6				
G12/FD	-	-	-				
JaCoP	658	-	1011156				
Gecode	-	60	-	71761	-	99	-	71186
or-tools	271	!	380	!	302	!	457	!
Opturion CPX	-	!	-	!	-	!	-	!
Choco3	-	-	-	-	-	-	-	-

Solver Time

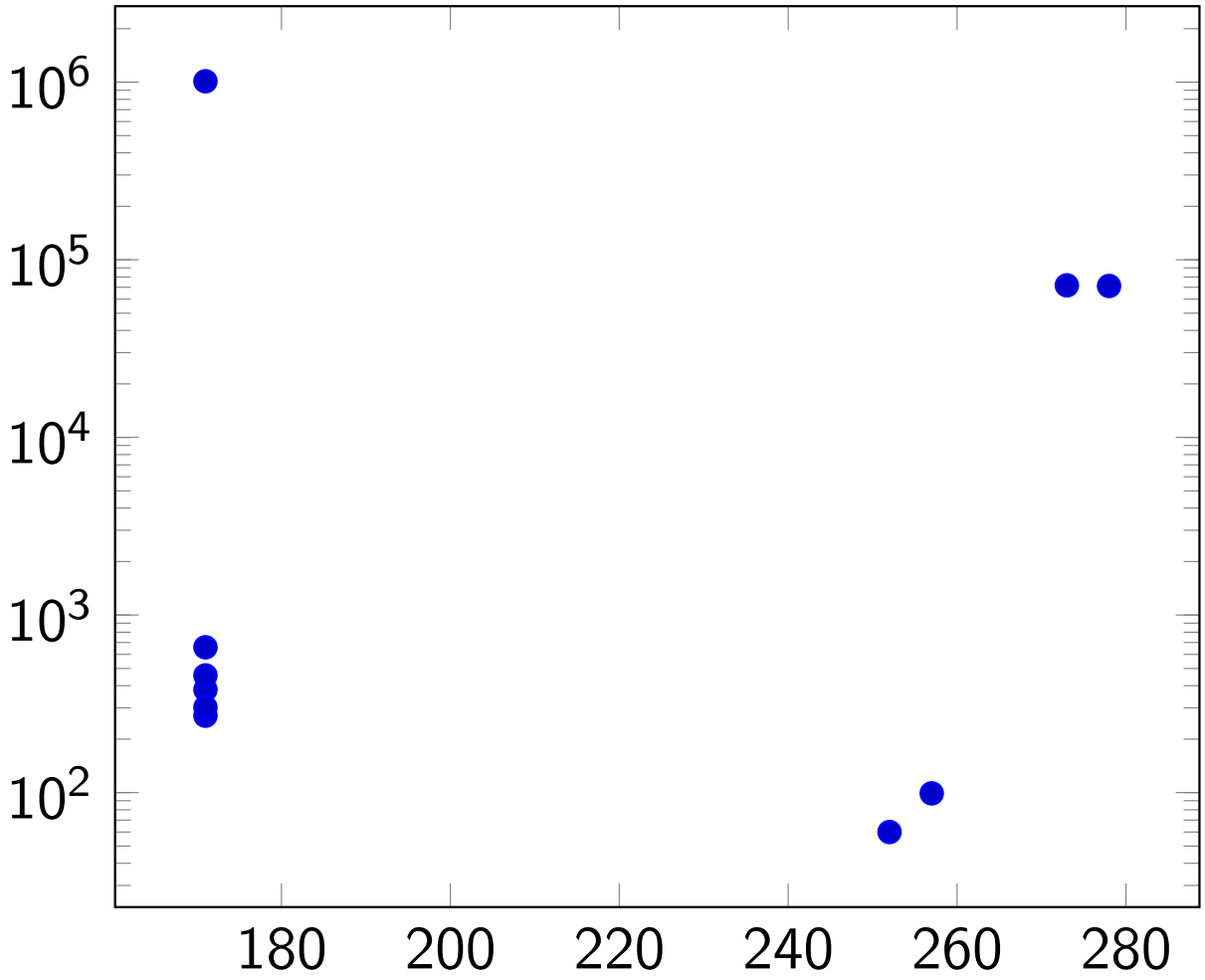
G12/FD								
JaCoP								
Gecode								
or-tools								
Opturion CPX								
Choco3	-	-	-	-	-	-	-	-



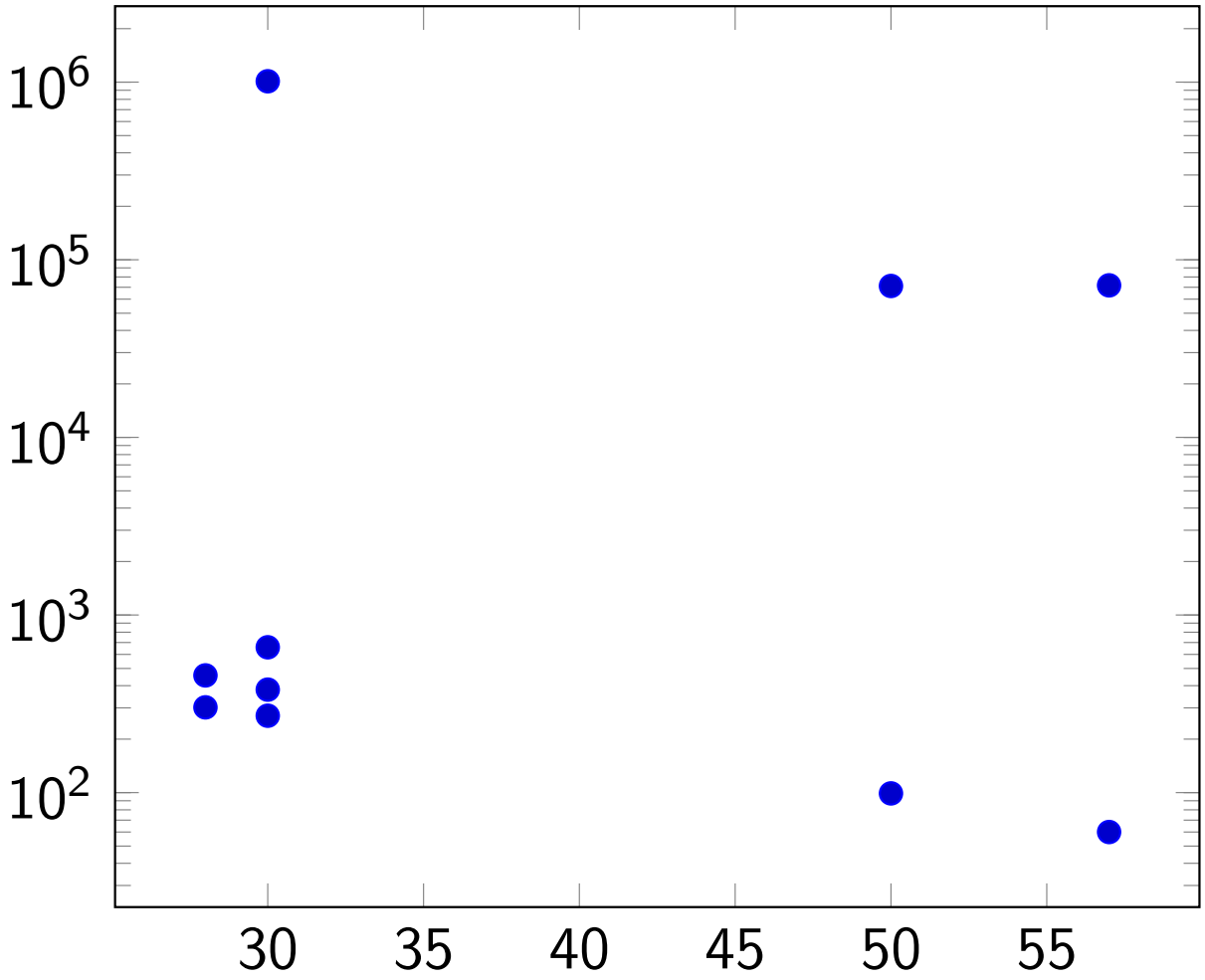
integer variables vs time



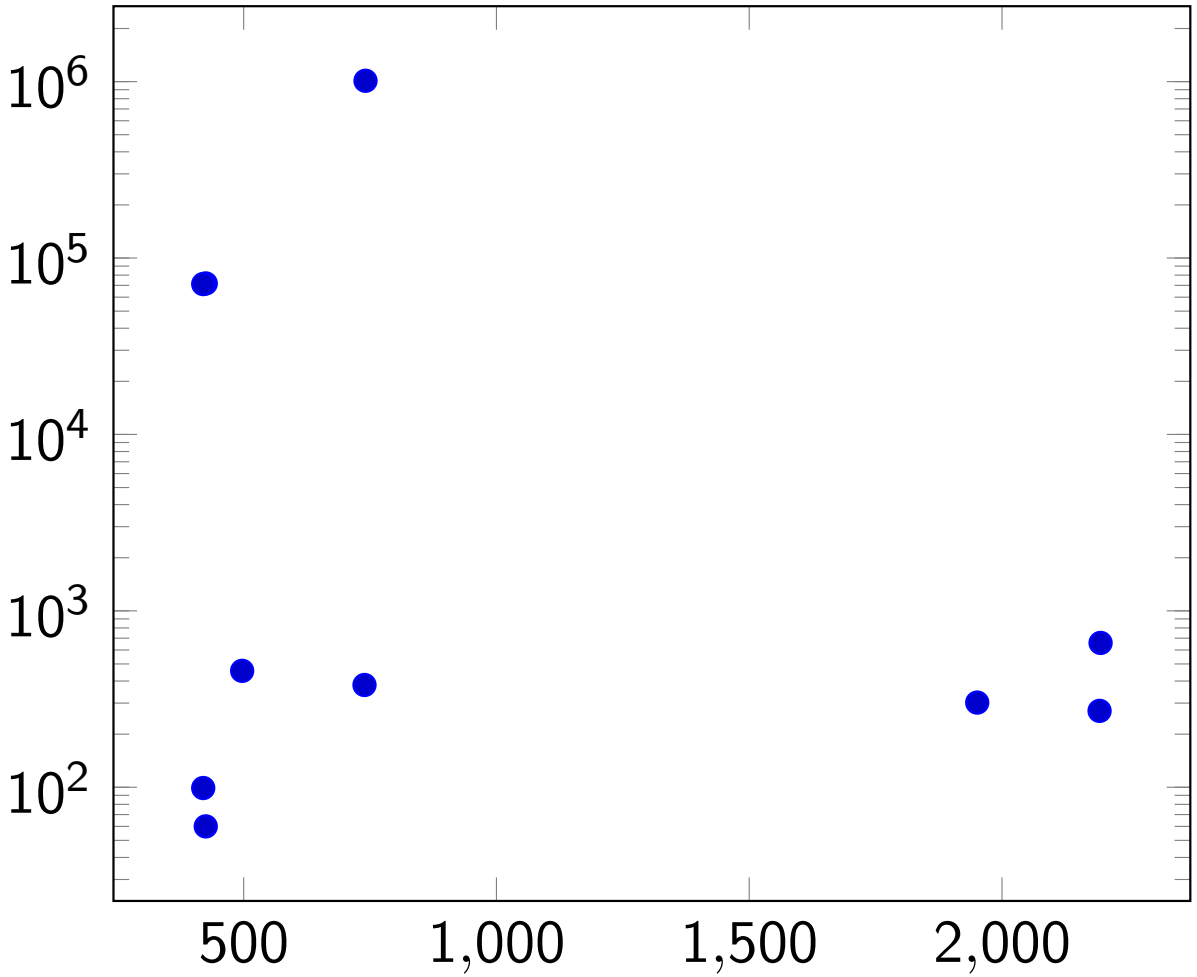
boolean variables vs time



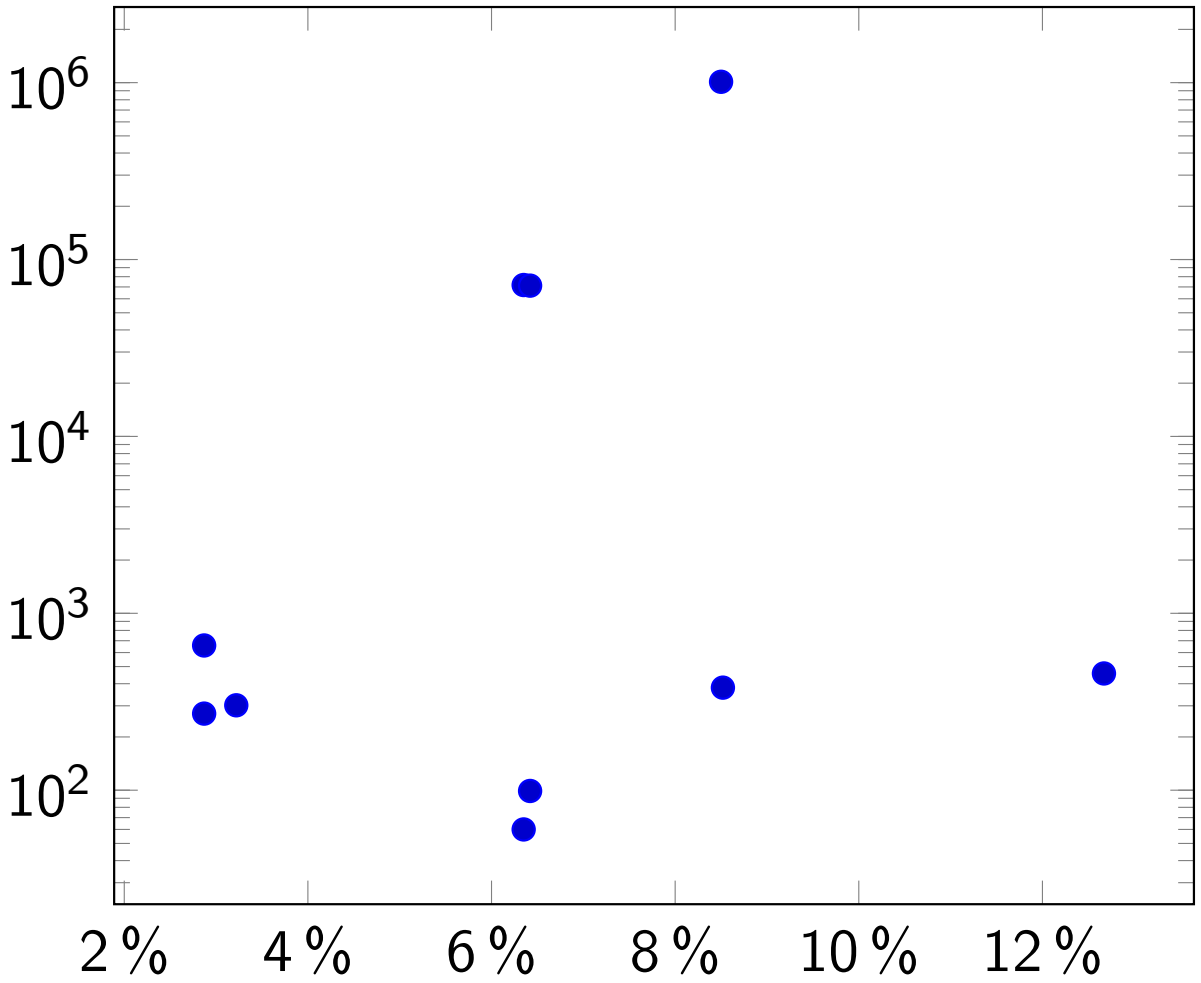
Total # of variables vs time



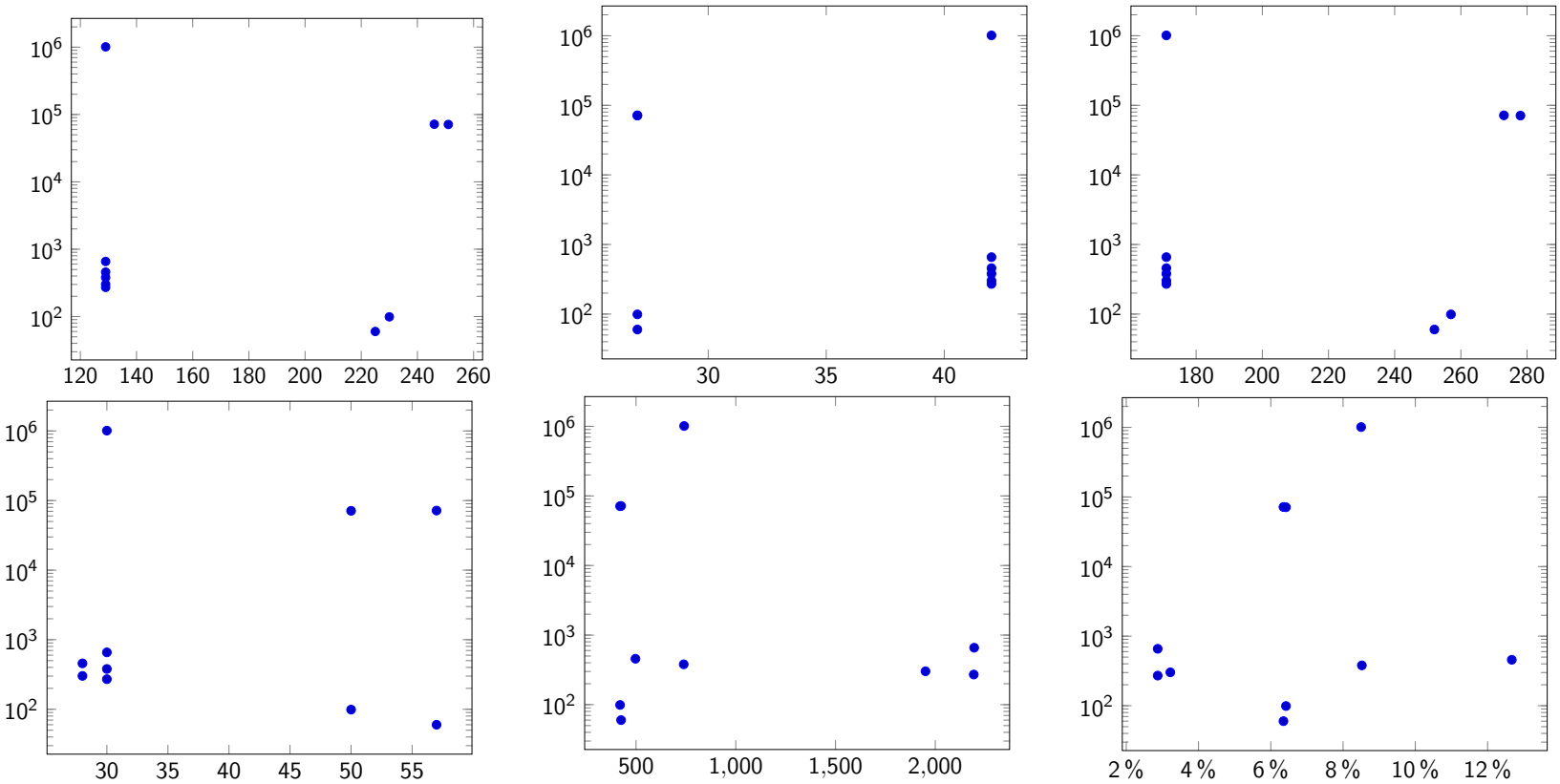
arrays vs time



constraints vs time



% reified vs time


10^6
10^5
10^4
10^3
10^2
120 140 160 180 200 220 240 260
30 35 40
180 200 220 240 260 280
30 35 40 45 50 55
500 1,000 1,500 2,000
2 % 4 % 6 % 8 % 10 % 12 %

Conclusions

Conclusions

- Model produces solution just as good as handmade solution

Conclusions

- Model produces solution just as good as handmade solution
- Solver performance varies a lot

Conclusions

- Model produces solution just as good as handmade solution
- Solver performance varies a lot
- Best performance: Gecode, all filters, MiniZinc 2.0.1

Conclusions

- Model produces solution just as good as handmade solution
- Solver performance varies a lot
- Best performance: Gecode, all filters, MiniZinc 2.0.1
- Filters presented have a positive impact on runtime

Conclusions

- Model produces solution just as good as handmade solution
- Solver performance varies a lot
- Best performance: Gecode, all filters, MiniZinc 2.0.1
- Filters presented have a positive impact on runtime
- No relation between FlatZinc output and solver runtime

Further work

- Test the result on a real robot
- Further testing of the filters
- More realistic representation of tools available
- Test solvers with more assemblies