The Art of Symmetry Breaking: Isomorph-Free Generation of Combinatorial Objects with SAT Modulo Symmetries

Stefan Szeider

Join work with Katalin Fazekas, Markus Kirchweger, Tomas Peitl, Manfred Scheucher, and Tianwei Zhang

NII SHONAN MEETING

FWF Der Wissenschaftsfonds.

WWTF VIENNA SCIENCE AND TECHNOLOGY FUND

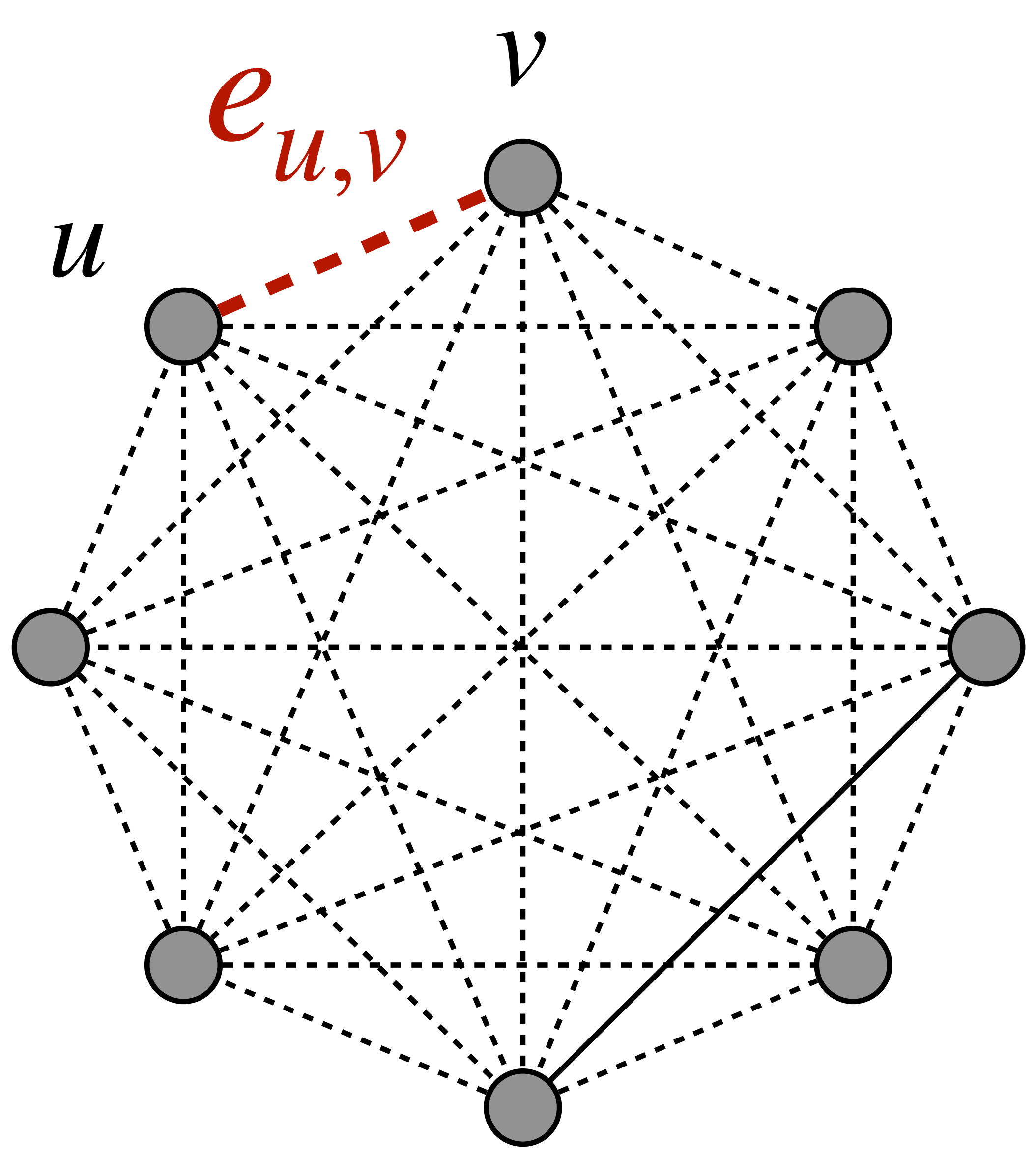
TU WIEN TECHNISCHE UNIVERSITÄT WIEN Vienna | Austria

ac ALGORITHMS AND COMPLEXITY GROUP

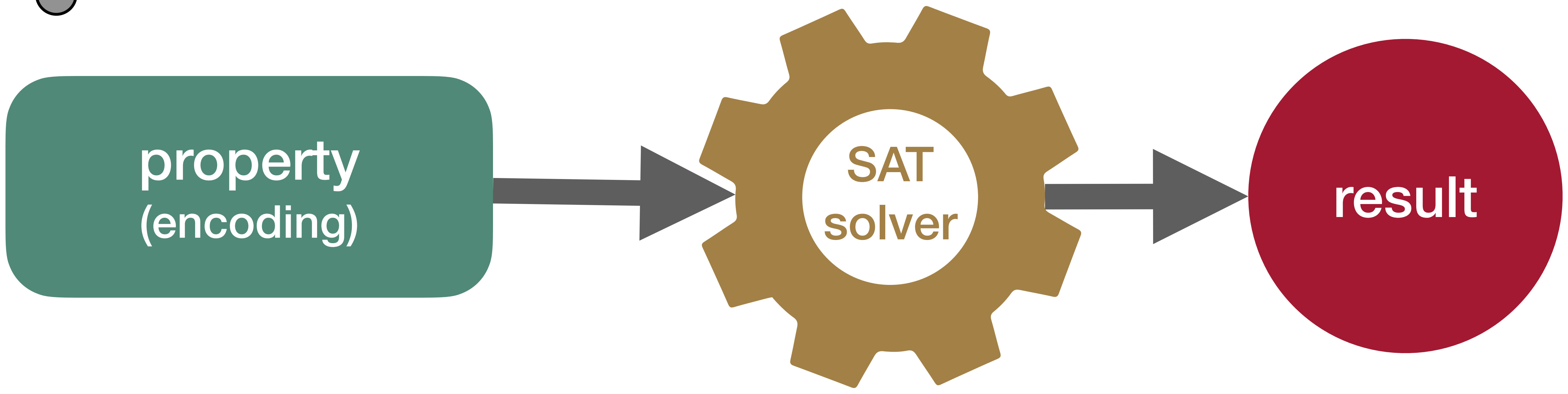
Generation of Combinatorial Objects

- Many problems in Discrete Mathematics ask for the (non-)existence of combinatorial objects with some desired property.
- **Combinatorial objects:** graphs, hypergraphs, matroids, etc.
- **Enumeration problems:** Enumerate all objects of size n with the property.
- **Extremal problems:** Graphs with smallest/largest number of edges and n vertices with the property?
- **Counterexamples to Conjectures:** Show that there is no object with the property of size up to n.

Formulate as a *Synthesis* Problem

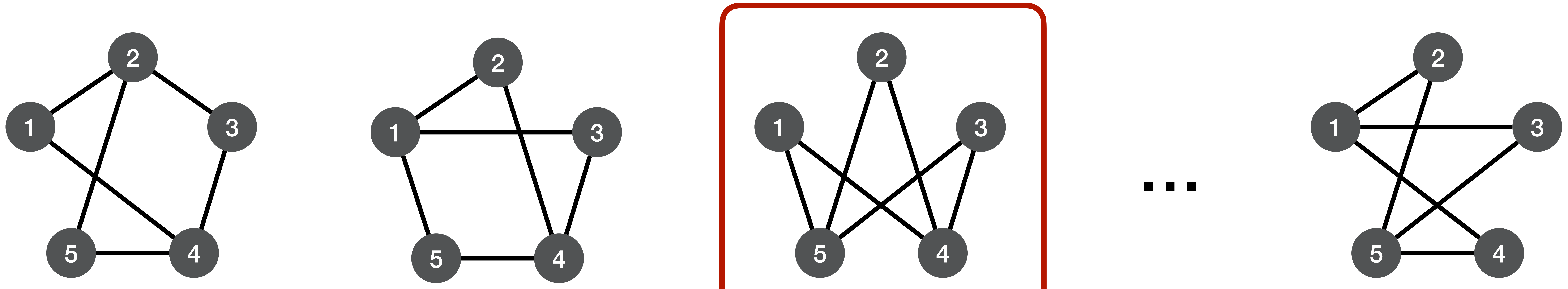


- We fix the number n of vertices, this gives $\binom{n}{2}$ many possible edges
- Each edge $\{u, v\}$ is represented by a variable $e_{u,v}$ which is true iff the edge exists



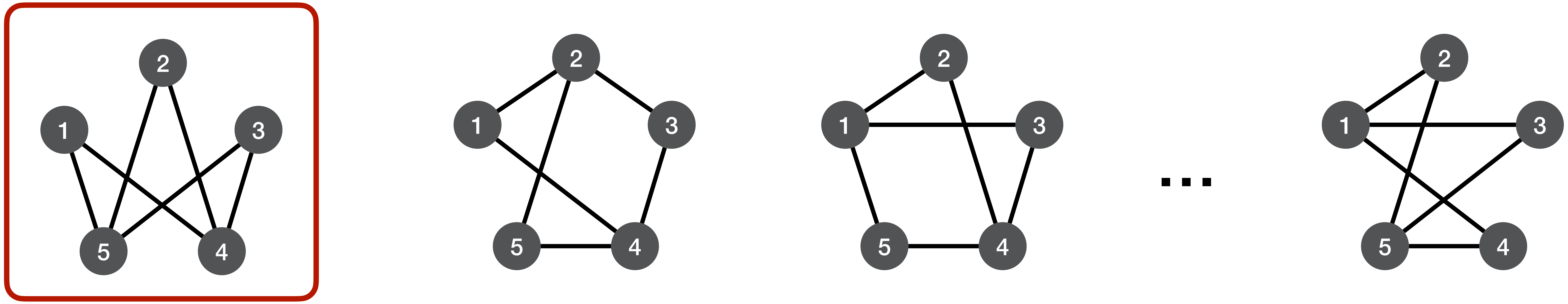
Isomorph-Free Generation

- **Isomorph-free generation:** Number of objects explode quickly
- **Canonization:** map each object G to a unique representative $\alpha(G)$ of its isomorphism class
- **Canonical Objects:** Only generate objects G with $\alpha(G) = G$



$\alpha(G_1) = G_3$ $\alpha(G_2) = G_3$ $\alpha(G_3) = G_3$ $\alpha(G_{120}) = G_3$

Canonical if has smallest adjacency matrix



0	0	0	1	1
0	0	0	1	1
0	0	0	1	1
1	1	1	0	0
1	1	1	0	0

$\prec$

0	1	0	1	0
1	0	1	0	1
0	1	0	1	0
1	0	1	0	1
0	1	0	1	0

$\prec$

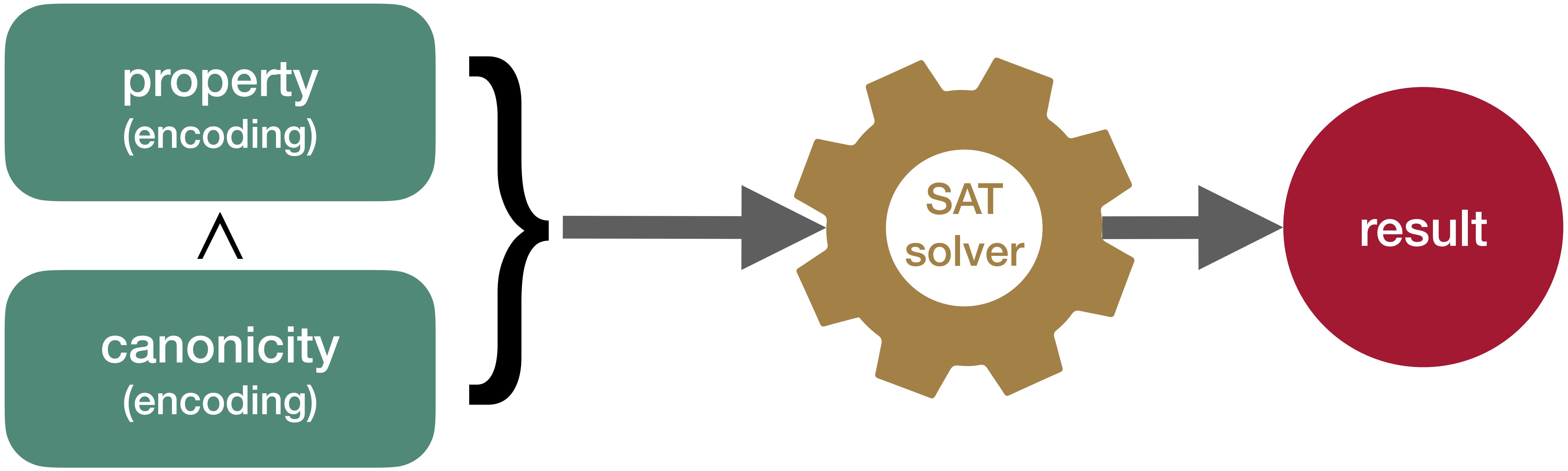
0	1	1	0	1
1	0	0	1	0
1	0	0	1	0
0	1	1	0	1
1	0	0	1	0

$\prec \dots \prec$

0	1	1	1	0
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
0	1	1	1	0

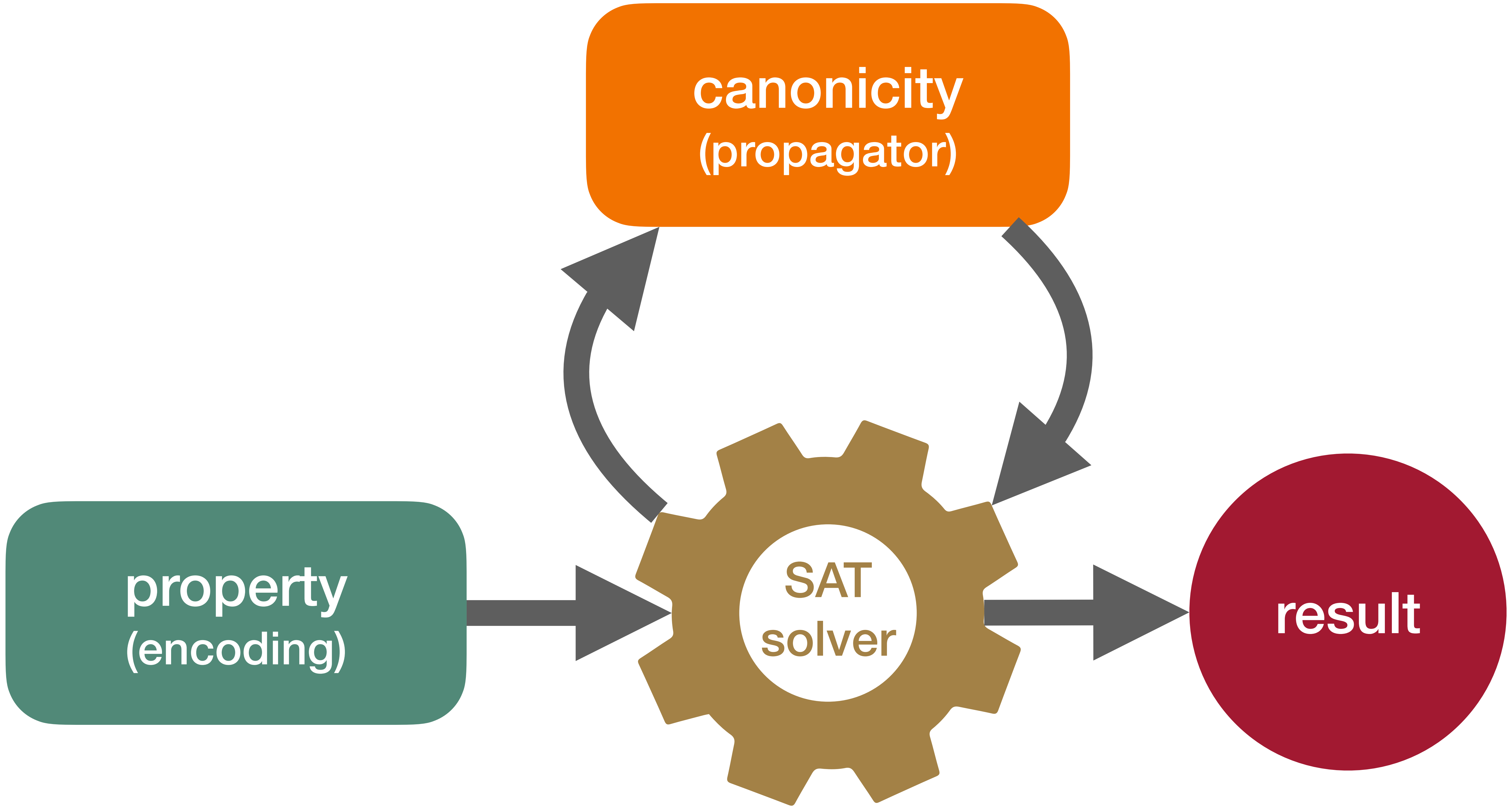
order adjacency matrices lexicographically (as strings obtained by concatenating rows)
declare the smallest one as canonical

Static SAT approach



- Problem: for canonicity no polynomial-size encoding is known
- Workaround: only partial symmetry breaking with a relaxed canonicity e.g., [Codish, Miller, Prosser, Stuckey, 2019]

Dynamic SAT approach



- CDCLSym, SAT+CAS, etc.

efficient canonicity test
also for directed graphs, multigraph,
edge-coloured graphs, matroids

partially defined graphs for early canonicity testing

[Kirchweger-Szeider CP'21]

SMS

SAT modulo Symmetries

[Fazekas-Niemetz-Preiner-Kirchweger-Szeider-Biere SAT '23]

communicate with SAT solver via IPASIR-UP

easy integration of other propagators

Boost library, planarity via Kuratowski
[Kirchweger-Scheucher-Szeider SAT'23]

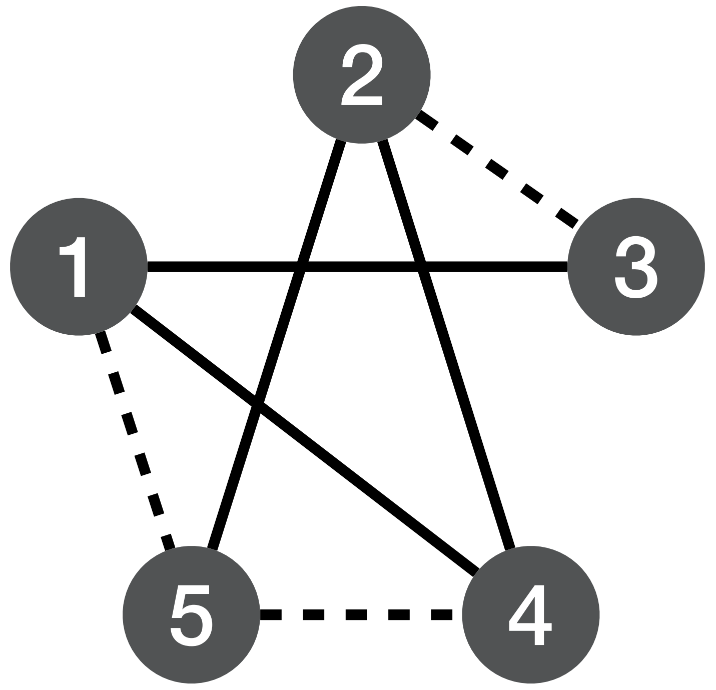
co-certificate learning, Kochen-Specker
[Kirchweger-Peitl-Szeider IJCAI'23]

Partially Defined Graphs

- some of the edges may be undecided whether they are present or not
- corresponding to a partial truth assignment to the edge variables

defined edge ———

undefined edge - - - - - - - -

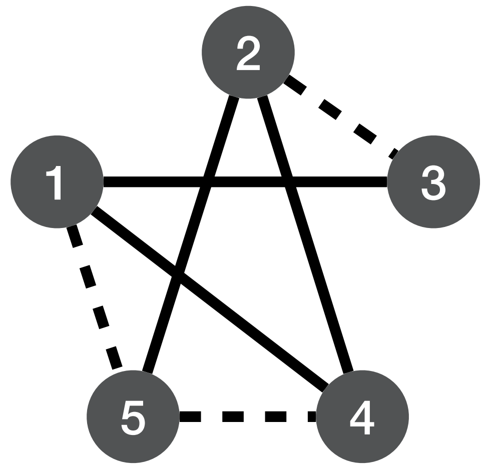


0	0	1	1	★
0	0	★	1	1
1	★	0	0	0
1	1	0	0	★
★	1	0	★	0

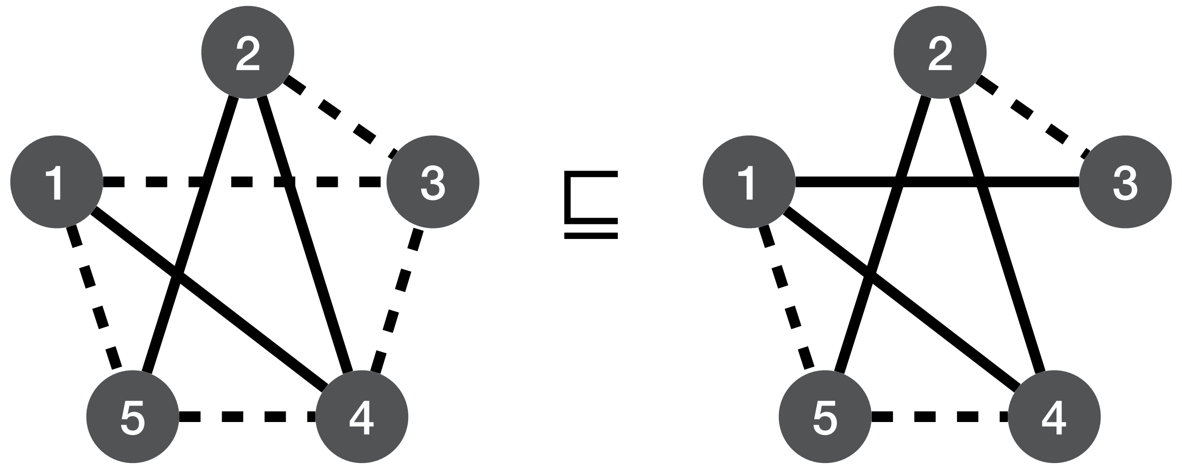
adjacency matrix

Extensions to fully defined graphs

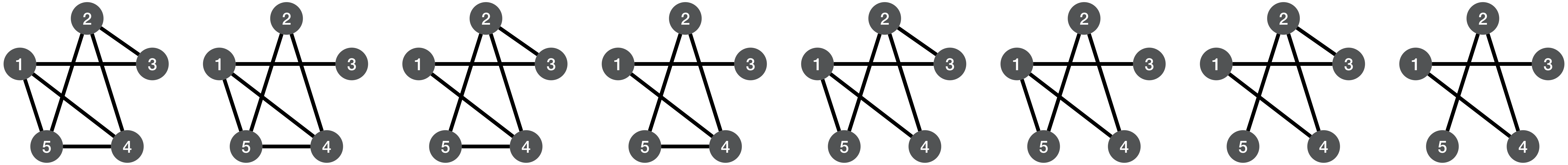
partially defined graph G



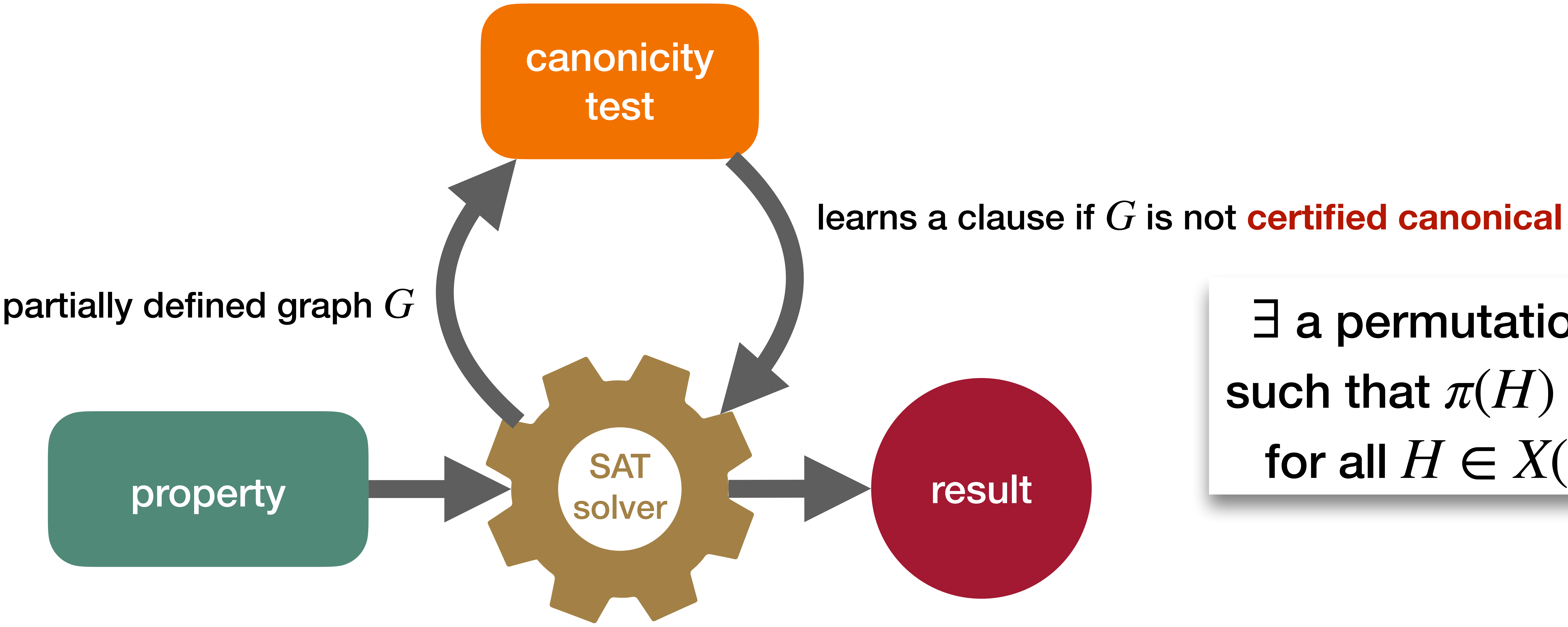
Partial Order: $G_1 \sqsubseteq G_2$ if $X(G_1) \supseteq X(G_2)$



$X(G)$: set of all fully defined graphs G can be extended to

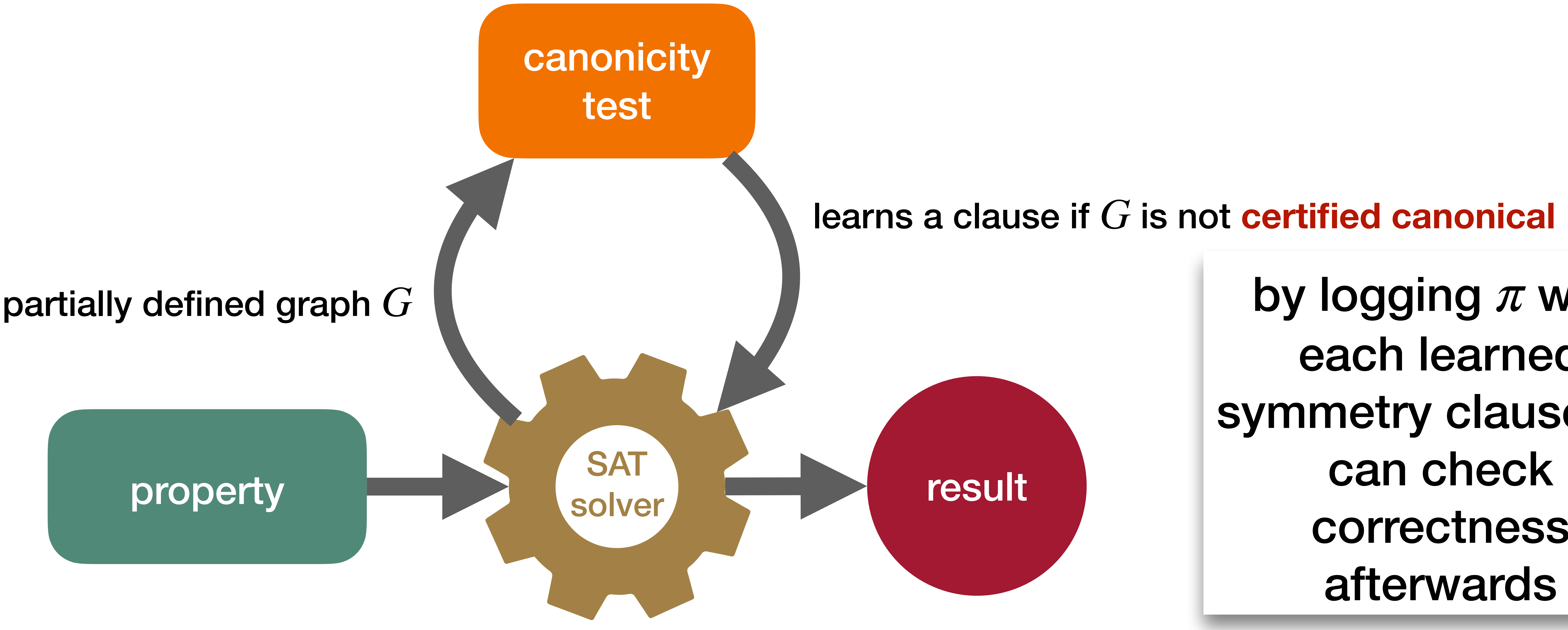


SMS canonicity test



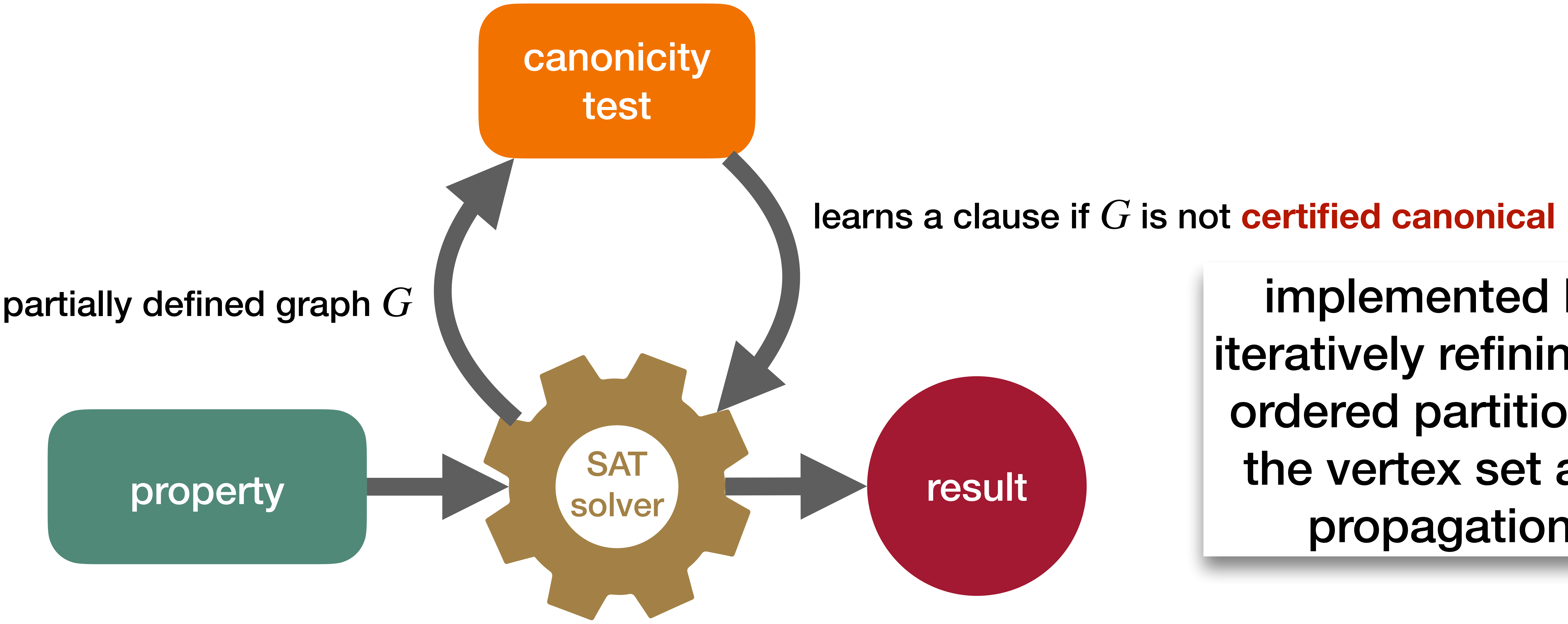
∃ a permutation π such that $\pi(H) \prec H$ for all $H \in X(G)$

SMS canonicity test



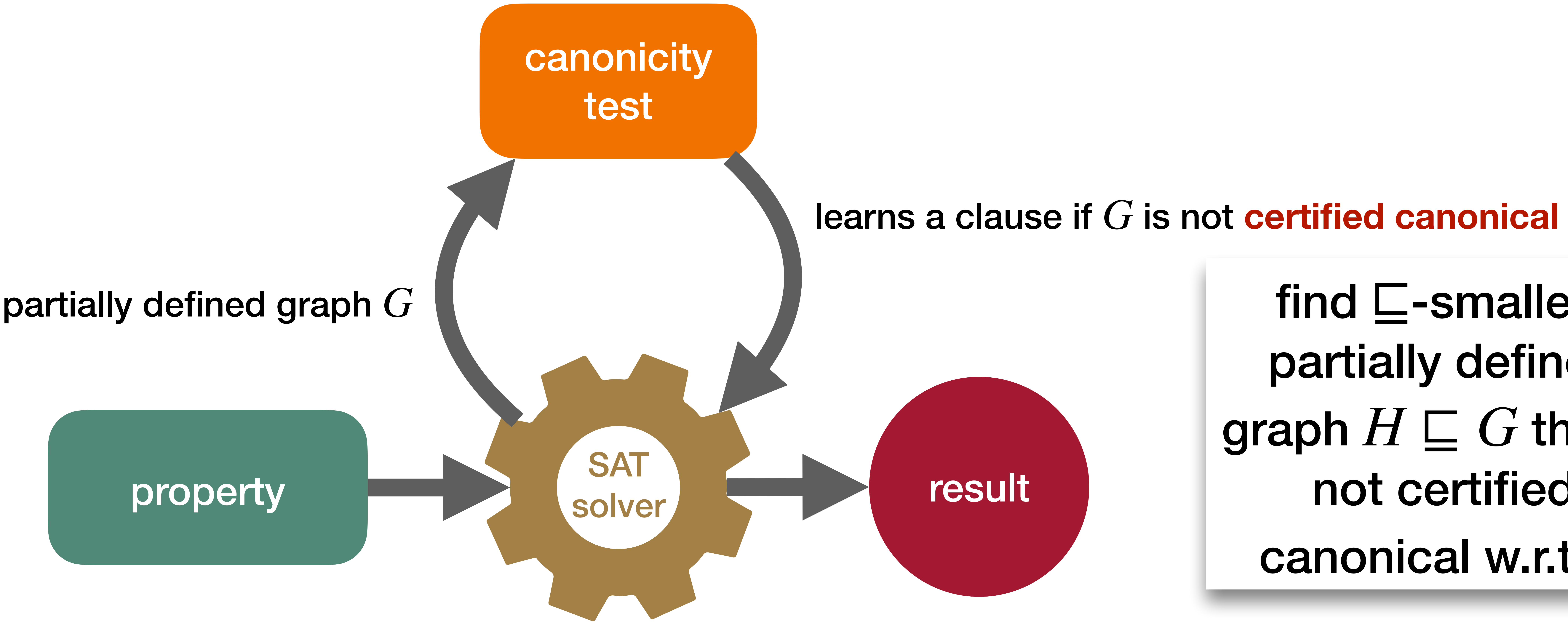
by logging π with each learned symmetry clause we can check correctness afterwards

SMS canonicity test



implemented by iteratively refining an ordered partition of the vertex set and propagation

SMS canonicity test



find $\sqsubseteq$-smallest partially defined graph $H \sqsubseteq G$ that is not certified canonical w.r.t. π

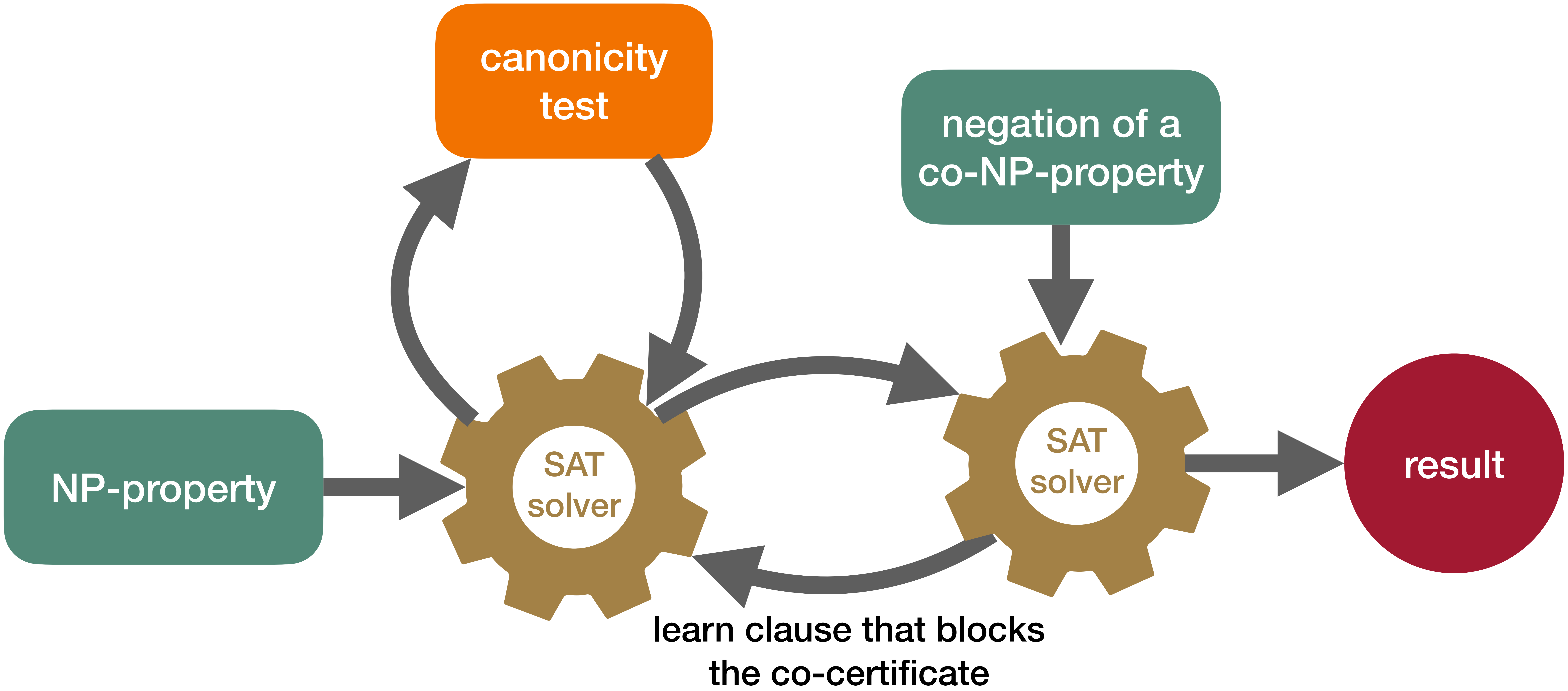
Co-NP Properties

triangle-free $\wedge$ not 3-colorable

NP	co-NP
encode with clauses	use a second SAT solver to check

[Kirchweger-Peitl-Szeider IJCAI'23]

SMS with co-certificate learning



Small number of co-certificates

n	△-free graphs	non-3 colorable	learned colorings
10	12.172	54	54
11	105.071	147	146
12	1.262.180	505	481
13	20.797.002	3.124	2.014
14	467.871.369	85.668	9.407

n=14: each co-certificate blocks 50k △-free graphs on average

Applications, Extensions, Results

[Kirchweger-Szeider CP'21]	**Simon-Murty Conjecture** on diameter-2 critical graphs, verified up to 18 vertices
[Kirchweger-Scheucher-Szeider SAT'22]	**Rota's Basis Conjecture** for matroids, DRAT proofs for SMS
[Kirchweger-Peitl-Szeider SAT'23]	**Erdős-Faber-Lovász Conjecture** for hypergraphs
[Kirchweger-Scheucher-Szeider SAT'23]	**Planar graphs and digraphs, enumeration**—planar Turán numbers, Earth-Moon problem, integer sequences for OEIS
[Kirchweger-Peitl-Szeider IJCAI'23]	**Co-certificate learning**—3D Kochen-Specker vector systems have at least 24 vectors
[Zhang-Szeider CP 2023]	**Universal graphs** and **universal tournaments**—templates, induced 7-universal graphs have at least 17 vertices

Summary

partially defined graphs for early canonicity testing

SMS
SAT modulo Symmetries

communicate with SAT solver via IPASIR-UP

easy integration of other propagators (e.g., Boost)

Tool https://github.com/markirch/sat-modulo-symmetries/

Documentation https://sat-modulo-symmetries.readthedocs.io/

Ressources

Tool https://github.com/markirch/sat-modulo-symmetries/

Documentation https://sat-modulo-symmetries.readthedocs.io/

Appendix: Code Examples

Basic usage of SMS

```
smsg -v 8
```

```
Number of vertices: 8
Initial partition:0 0 0 0 0 0 0 0
Clauses: 0, Variables 28
SAT Solver: Cadical
Starting to solve
Solution 1
[(0,1),(0,2),(0,3),(0,4),(0,5),(0,6),(0,7),(1,2),(1,3),(1,4),
(1,5),(1,6),(1,7),(2,3),(2,4),(2,5),(2,6),(2,7),(3,4),(3,5),
(3,6),(3,7),(4,5),(4,6),(4,7),(5,6),(5,7),(6,7)]
Search finished
Total time: 0.032886
```

```
smsg -v 8 --all-graphs --print-stats
```

```
[…]
Solution 12346
[(0,6),(0,7),(1,5),(1,6),(1,7),(2,4),(2,5),(3,4),(3,5),(3,6),(3,7),
(4,5),(4,7),(5,6),(5,7)]
c falsified clause is learnt from external propagator
Search finished
Number of solutions: 12346
Time in propagator: 0.022616
Time in check full graphs: 0.003173
Calls of check: 12346
Calls propagator: 27891
Statistics for MinimalityChecker:
        Calls: 13391
        Time in seconds: 0.3375
        Added clauses: 1044
Total time: 0.625818
```

Python wrapper

```
python pysms/graph_builder.py -v 5
```

```
Arguments: Namespace(vertices=5, cnf_file=None, directed=False,
multigraph=None, underlying_graph=False, static_partition=False,
counter='sequential', DEBUG=1, all_graphs=False, hide_graphs=False,
args_SMS='', num_edges_upp=None, num_edges_low=None, Delta_upp=None,
delta_low=None, even_degrees=False,
no_subsuming_neighborhoods=False, degree_partition=False,
chi_upp=None, chi_low=None, Ck_free=None, mtf=False, girth=None,
girth_compact=None, alpha_upp=None, omega_upp=None, ramsey=None,
planar_kuratowski=False, connectivity_low=0, diam2_critical=False)
running the command:  time smsg --vertices 5 --print-stats True --
frequency 30  --dimacs ./temp2062.enc
[…]
```

```
[…]
Number of vertices: 5
Initial partition:0 0 0 0 0
Clauses: 0, Variables 10
SAT Solver: Cadical
Starting to solve
Solution 1
[(0,1),(0,2),(0,3),(0,4),(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)]
Search finished
Time in propagator: 0.000003
Time in check full graphs: 0.000000
Calls of check: 1
Calls propagator: 11
Statistics for MinimalityChecker:
        Calls: 1
        Time in seconds: 0.0001
        Added clauses: 0
Total time: 0.000303
```

Example: Triangle-free graphs with minimum degree at least 3

```
python pysms/graph_builder.py -v 10 -a --Ck-free 3 --delta-low 2
```

```
[…]
Solution 3494
[(0,8),(0,9),(1,7),(1,9),(2,7),(2,8),(3,6),(3,8),(3,9),(4,6),(4,8),(4,9),
(5,6),(5,8),(5,9)]
Search finished
Number of solutions: 3494
Time in propagator: 0.009958
Time in check full graphs: 0.000889
Calls of check: 3494
Calls propagator: 11580
Statistics for MinimalityChecker:
        Calls: 4738
        Time in seconds: 0.4555
        Added clauses: 1244
Total time: 0.589939
```

Planar graphs

```
python pysms/graph_builder.py -v 10 -a --planar
                                --Ck-free 3 --delta-low 2
```

```
[…]
Solution 1478
[(0,8),(0,9),(1,6),(1,7),(2,4),(2,5),(2,8),(2,9),(3,4),(3,5),(3,8),(3,9),(4,6),(4,7),(5,6),
(5,7)]
Search finished
Number of solutions: 1478
Time in propagator: 0.026903
Time in check full graphs: 0.000389
Calls of check: 1478
Calls propagator: 8383
Statistics for MinimalityChecker:
        Calls: 2760
        Time in seconds: 0.2729
        Added clauses: 912
Statistics for PlanarityChecker:
        Calls: 2871
        Time in seconds: 0.0527
        Added clauses: 674
Total time: 0.412692
```

Other options

```
usage: graph_builder.py [-h] --vertices VERTICES [--cnf-file CNF_FILE]
[--directed] [--multigraph MULTIGRAPH] [--underlying-graph]
                        [--static-partition] [--counter
{sequential,totalizer}] [--DEBUG DEBUG] [--all-graphs] [--hide-graphs]
                        [--args-SMS ARGS_SMS] [--num-edges-upp
NUM_EDGES_UPP] [--num-edges-low NUM_EDGES_LOW] [--Delta-upp DELTA_UPP]
                        [--delta-low DELTA_LOW] [--even-degrees] [--
no-subsuming-neighborhoods] [--degree-partition]
                        [--chi-upp CHI_UPP] [--chi-low CHI_LOW] [--Ck-
free CK_FREE] [--mtf] [--girth GIRTH]
                        [--girth-compact GIRTH_COMPACT] [--alpha-upp
ALPHA_UPP] [--omega-upp OMEGA_UPP] [--ramsey RAMSEY RAMSEY]
                        [--planar-kuratowski] [--connectivity-low
CONNECTIVITY_LOW] [--diam2-critical]
```

```
optional arguments:
  -h, --help            show this help message and exit

Main arguments:
  The number of vertices is mandatory, everything else is optional

  --vertices VERTICES, -v VERTICES
                        number of vertices
  --cnf-file CNF_FILE   store the generated encoding here
  --directed, -d        search for directed graphs
  --multigraph MULTIGRAPH, -m MULTIGRAPH
                        search for a multigraph
  --underlying-graph    consider the underlying undirected graph for
directed graphs
  --static-partition    specify a statically enforced partial vertex
ordering (respected by SMS)
  --counter {sequential,totalizer}
                        the CNF encoding for cardinality constraints
  --DEBUG DEBUG, -D DEBUG
                        debug level
```

```
Solver options:
  --all-graphs, -a      generate all graphs (without this, the solver
will exit after the first solution)
  --hide-graphs, -hg    do not display graphs (meant as a counting
functionality, though the graphs still need to be enumerated)
  --args-SMS ARGS_SMS   command line to be appended to the call to
smsg/smsd (see src/main.cpp or README.md)
```

Pre-defined encodings

```
Graph constraints:
  A set of pre-defined constraints for common applications, including
applications from SMS papers

  --num-edges-upp NUM_EDGES_UPP
                        upper bound on the maximum number of edges
  --num-edges-low NUM_EDGES_LOW
                        lower bound on the minimum number of edges
  --Delta-upp DELTA_UPP
                        upper bound on the maximum degree
  --delta-low DELTA_LOW
                        lower bound on the minimum degree
  --even-degrees        all degrees should be even
  --no-subsuming-neighborhoods
                        ensure that N(v) ⊈ N(u) for any vertex pair u,v
  --degree-partition    sort vertices by degree and only apply SMS on
vertices with same degree
  --chi-upp CHI_UPP     upper bound on the chromatic number
  --chi-low CHI_LOW     lower bound on the chromatic number (not encoded
to CNF, needs SMS)
```

```
 --Ck-free CK_FREE     forbid the k-cycle C_k as (non-induced)
subgraph
  --mtf                 search for maximal triangle-free graphs (where
adding any edge creates a triangle)
  --girth GIRTH         lower bound on girth
  --girth-compact GIRTH_COMPACT
                        lower bound on girth, more compact encoding
  --alpha-upp ALPHA_UPP
                        maximum size of an independent set
  --omega-upp OMEGA_UPP
                        maximum size of a clique
  --ramsey RAMSEY RAMSEY
                        (a, w) means no independent set of size a and
no clique of size w
  --planar-kuratowski, --planar, -p
                        generate only planar graphs (not encoded to
CNF, needs SMS)
  --connectivity-low CONNECTIVITY_LOW, -c CONNECTIVITY_LOW, --kappa-
low CONNECTIVITY_LOW
                        lower bound on vertex connectivity
  --diam2-critical      assert a diameter-2-critical graph
```

Building your own encoding

- Example: triangle-free graphs

```
from pysms.graph_builder import GraphEncodingBuilder
from itertools import combinations
builder = GraphEncodingBuilder(7, directed=False)
for i,j,k in combinations(builder.V, 3):
    builder.append([-builder.var_edge(i,j), -builder.var_edge(i,k),
-builder.var_edge(j,k)])
builder.solve(allGraphs=True)
```