How to read the trace of the solver

Let us explain the trace of the solver for `isaplanner_prop31.smt2`, i.e., trying to prove the property $\forall\ (a : nat)\ (b : nat)\ (c : nat).\ (min\ (min\ a\ b)\ c) = (min\ a\ (min\ b\ c))$. The first section of the trace recalls the parameters that were used for the model-inference: *ICE fuel* which states the maximal number of ICE iterations, *Timeout* the maximal timeout, and *Convolution* the type of convolution used for the proof, i.e., either left, right or complete.

```
Inference procedure has parameters:
Ice fuel: 200
Timeout: 60s
Convolution: right
```

The second section recalls the clauses defining the program and the clauses representing the properties.

```
Learning problem is:
env: {
nat -> {s, z}
}
definition:
{
(min, F:
{() -> min([s(u), z, z])
() -> min([z, y, z])
(min([u, y1, _qb])) -> min([s(u), s(y1), s(_qb)])}
(min([_rb, _sb, _tb]) /\ min([_rb, _sb, _ub])) -> eq_nat([_tb, _ub])
)
}

properties:
{(min([_vb, c, _wb]) /\ min([a, _xb, _yb]) /\ min([a, b, _vb]) /\ min([b, c, _xb])) ->
    eq_nat([_wb, _yb])}
```

The following section states which relation can be over/under-approximated.

```
over-approximation: {min}
under-approximation: {eq_nat}

Clause system for inference is:

{
() -> min([s(u), z, z]) -> 0
() -> min([z, y, z]) -> 0
(min([_vb, c, _wb]) /\ min([a, _xb, _yb]) /\ min([a, b, _vb]) /\ min([b, c, _xb])) ->
    eq_nat([_wb, _yb]) -> 0
(min([u, y1, _qb])) -> min([s(u), s(y1), s(_qb)]) -> 0
}
```

Finally, the total time is given. If the proof is successful, the solver provides the model, i.e., the convoluted tree automata for all the relations are stated. The solver only provides automata for non-equality relations, here *min*, as automata for equality on any datatype are canonical.

```
Solving took 0.068141 seconds.
Proved
Model:
|_
{
min ->
{{{
Q={q_gen_432, q_gen_434},
Q_f={q_gen_432},
Delta=
{
<s>(q_gen_434) -> q_gen_434
<z>() -> q_gen_434
<s, s, s>(q_gen_432) -> q_gen_432
<s, z, z>(q_gen_434) -> q_gen_432
<z, s, z>(q_gen_434) -> q_gen_432
<z, z, z>() -> q_gen_432
}

Datatype: <nat, nat, nat>
Convolution form: right
}}}
}
```

Note that the convoluted automata produced by our implementation are padding-free. Ruling out padding symbols allows to avoid having different states recognizing similar relations. For instance, symbols $\langle \square, \square, Z \rangle$, $\langle \square, Z, \square \rangle$ and $\langle Z, \square, \square \rangle$ are different but they all represent the unary tuple (Z). Making no difference between those three relations is safe w.r.t. the recognized language and only needs one state instead of three.