Validation of QBF Encodings with Winning Strategies

Irfansha Shaik [1] Maximilian Heisinger [2] Martina Seidl [2] Jaco van de Pol [1]

July 8, 2023

[1] Aarhus University, Denmark

[2] Johannes Kepler Universität Linz, Austria

Challenges in Correctness of QBF Modelling

- Problems like Planning and 2-player games can be encoded using SAT

Challenges in Correctness of QBF Modelling

- Problems like Planning and 2-player games can be encoded using SAT
- Same problems could be encoded more compactly using QBF

Challenges in Correctness of QBF Modelling

- Problems like Planning and 2-player games can be encoded using SAT
- Same problems could be encoded more compactly using QBF
- Unlike SAT, a complete assignment of inner variables is non-trivial in QBF

Challenges in Correctness of QBF Modelling

- Problems like Planning and 2-player games can be encoded using SAT
- Same problems could be encoded more compactly using QBF
- Unlike SAT, a complete assignment of inner variables is non-trivial in QBF

Challenges in Correctness of QBF Modelling

- Problems like Planning and 2-player games can be encoded using SAT
- Same problems could be encoded more compactly using QBF
- Unlike SAT, a complete assignment of inner variables is non-trivial in QBF

What can go wrong?

- Solver could be buggy

Challenges in Correctness of QBF Modelling

- Problems like Planning and 2-player games can be encoded using SAT
- Same problems could be encoded more compactly using QBF
- Unlike SAT, a complete assignment of inner variables is non-trivial in QBF

What can go wrong?

- Solver could be buggy
- **Encoding can be incorrect**

Challenges in Correctness of QBF Modelling

- Problems like Planning and 2-player games can be encoded using SAT
- Same problems could be encoded more compactly using QBF
- Unlike SAT, a complete assignment of inner variables is non-trivial in QBF

What can go wrong?

- Solver could be buggy
- **Encoding can be incorrect**

Validation Challenges

- Is the problem encoded correctly?

Challenges in Correctness of QBF Modelling

- Problems like Planning and 2-player games can be encoded using SAT
- Same problems could be encoded more compactly using QBF
- Unlike SAT, a complete assignment of inner variables is non-trivial in QBF

What can go wrong?

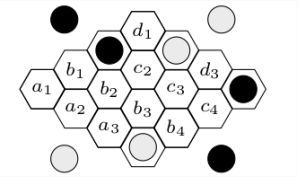
- Solver could be buggy
- **Encoding can be incorrect**

Validation Challenges

- Is the problem encoded correctly?
- Do encoding variations preserve correctness?

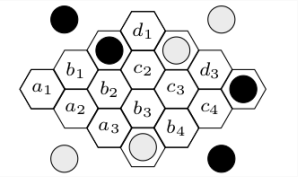
A case study: Hex encoding

Hein Puzzle 12



A case study: Hex encoding

Hein Puzzle 12



Structure

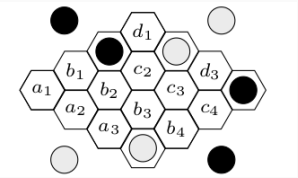
$\exists \mathsf{M}^1 \forall \mathsf{M}^2 \ldots \exists \mathsf{M}^d$

$\exists \mathsf{W}_1, \ldots, \mathsf{W}_\mathsf{k}$

...

A case study: Hex encoding

Hein Puzzle 12



Encoding Variations

- LN : both players can only occupy empty cells

Structure

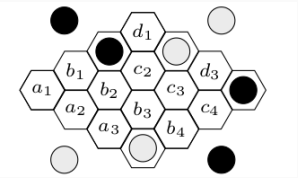
$\exists \mathsf{M}^1 \forall \mathsf{M}^2 \ldots \exists \mathsf{M}^d$

$\exists \mathsf{W}_1, \ldots, \mathsf{W}_\mathsf{k}$

...

A case study: Hex encoding

Hein Puzzle 12



Encoding Variations

- LN : both players can only occupy empty cells
- SN : black player can only occupy empty cells

Structure

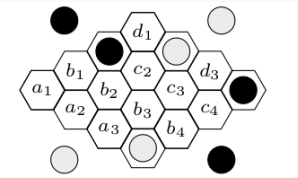
$\exists \mathsf{M}^1 \forall \mathsf{M}^2 \ldots \exists \mathsf{M}^d$

$\exists \mathsf{W}_1, \ldots, \mathsf{W}_\mathsf{k}$

...

A case study: Hex encoding

Hein Puzzle 12



Encoding Variations

- LN : both players can only occupy empty cells
- SN : black player can only occupy empty cells
- SN-R : black player can occupy empty/black cells

Structure

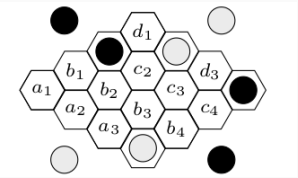
$\exists \mathsf{M}^1 \forall \mathsf{M}^2 \ldots \exists \mathsf{M}^d$

$\exists \mathsf{W}_1, \ldots, \mathsf{W}_k$

...

A case study: Hex encoding

Hein Puzzle 12



Encoding Variations

- LN : both players can only occupy empty cells
- SN : black player can only occupy empty cells
- SN-R : black player can occupy empty/black cells

Structure

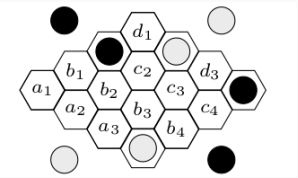
$\exists \mathsf{M}^1 \forall \mathsf{M}^2 \ldots \exists \mathsf{M}^d$

$\exists \mathsf{W}_1, \ldots, \mathsf{W}_k$

...

A case study: Hex encoding

Hein Puzzle 12



Encoding Variations

- LN : both players can only occupy empty cells
- SN : black player can only occupy empty cells
- SN-R : black player can occupy empty/black cells

Move variables

```
e 1 2 3 0
a 4 5 6 0
e 7 8 9 0
a 10 11 12 0
e 13 14 15 0
a 16 17 18 0
e 19 20 21 0
```

Structure

$\exists \mathsf{M}^1 \forall \mathsf{M}^2 \ldots \exists \mathsf{M}^d$

$\exists \mathsf{W}_1, \ldots, \mathsf{W}_k$

...

Validation of QBF Encodings

Solution 1: Interactive Play + Assertions for Validation

- Interactive play allows extracting assignments of inner variables in QBF

Solution 1: Interactive Play + Assertions for Validation

- Interactive play allows extracting assignments of inner variables in QBF
- The main idea is to add assertions in interactive play for validation

Solution 1: Interactive Play + Assertions for Validation

- Interactive play allows extracting assignments of inner variables in QBF
- The main idea is to add assertions in interactive play for validation

Solution 1: Interactive Play + Assertions for Validation

- Interactive play allows extracting assignments of inner variables in QBF
- The main idea is to add assertions in interactive play for validation

```
e 1 2 3 0
a 4 5 6 0
e 7 8 9 0
a 10 11 12 0
e 13 14 15 0
a 16 17 18 0
e 19 20 21 0
```

Move variables

Solution 1: Interactive Play + Assertions for Validation

- Interactive play allows extracting assignments of inner variables in QBF
- The main idea is to add assertions in interactive play for validation

```
e 1 2 3 0
a 4 5 6 0
e 7 8 9 0
a 10 11 12 0
e 13 14 15 0
a 16 17 18 0
e 19 20 21 0
```

Move variables

Interactive play with Skolem/Herbrand Functions

- Skolem Function: $\forall \mapsto \exists$ and Herbrand Function: $\exists \mapsto \forall$

Solution 1: Interactive Play + Assertions for Validation

- Interactive play allows extracting assignments of inner variables in QBF
- The main idea is to add assertions in interactive play for validation

```
e 1 2 3 0
a 4 5 6 0
e 7 8 9 0
a 10 11 12 0
e 13 14 15 0
a 16 17 18 0
e 19 20 21 0
```

Move variables

Interactive play with Skolem/Herbrand Functions

- Skolem Function: $\forall \mapsto \exists$ and Herbrand Function: $\exists \mapsto \forall$
- Such functions can be extracted from certifying solvers

Solution 1: Interactive Play + Assertions for Validation

- Interactive play allows extracting assignments of inner variables in QBF
- The main idea is to add assertions in interactive play for validation

```
e 1 2 3 0
a 4 5 6 0
e 7 8 9 0
a 10 11 12 0
e 13 14 15 0
a 16 17 18 0
e 19 20 21 0
```

Move variables

Interactive play with Skolem/Herbrand Functions

- Skolem Function: $\forall \mapsto \exists$ and Herbrand Function: $\exists \mapsto \forall$
- Such functions can be extracted from certifying solvers

Solution 1: Interactive Play + Assertions for Validation

- Interactive play allows extracting assignments of inner variables in QBF
- The main idea is to add assertions in interactive play for validation

```
e 1 2 3 0
a 4 5 6 0
e 7 8 9 0
a 10 11 12 0
e 13 14 15 0
a 16 17 18 0
e 19 20 21 0
```

Move variables

Interactive play with Skolem/Herbrand Functions

- Skolem Function: $\forall \mapsto \exists$ and Herbrand Function: $\exists \mapsto \forall$
- Such functions can be extracted from certifying solvers

Interactive play with a QBF solver

- All QBF solvers provide outermost assignment.

Solution 1: Interactive Play + Assertions for Validation

- Interactive play allows extracting assignments of inner variables in QBF
- The main idea is to add assertions in interactive play for validation

```
e 1 2 3 0
a 4 5 6 0
e 7 8 9 0
a 10 11 12 0
e 13 14 15 0
a 16 17 18 0
e 19 20 21 0
```

Move variables

Interactive play with Skolem/Herbrand Functions

- Skolem Function: $\forall \mapsto \exists$ and Herbrand Function: $\exists \mapsto \forall$
- Such functions can be extracted from certifying solvers

Interactive play with a QBF solver

- All QBF solvers provide outermost assignment.
- Extract inner variable assignments from modified QBFs

An Outline for Interactive Play

Pseudocode for True instance validation

1: **for** $i = 1;\ i < \max;\ i = i + 2$ **do**
2: Extract $\exists$-move
3: Input $\forall$-move
4: **end for**
5: Check the Assertion

$\exists\, M^1$

$\forall\, M^2$

$\exists\, M^3$

$\ldots$

$\exists\, M^d$

$\exists\, W_1, \ldots, W_k$

...

An Outline for Interactive Play

Pseudocode for True instance validation

1: **for** $i = 1;\ i < \max;\ i = i + 2$ **do**
2: **Extract** $\exists$-move : M^1
3: Input $\forall$-move
4: **end for**
5: Check the Assertion

$\exists\, \mathbf{M}^1$

$\forall\, \mathrm{M}^2$

$\exists\, \mathrm{M}^3$

...

$\exists\, \mathrm{M}^d$

$\exists\, \mathrm{W}_1, \ldots, \mathrm{W}_k$

...

An Outline for Interactive Play

Pseudocode for True instance validation

1: **for** $i = 1;\ i < \max;\ i = i + 2$ **do**
2: Extract $\exists$-move
3: **Input** $\forall$-move : M^2
4: **end for**
5: Check the Assertion

$\exists\, \mathbf{M}^1$

$\forall\, \mathbf{M}^2$

$\exists\, \mathrm{M}^3$

...

$\exists\, \mathrm{M}^d$

$\exists\, \mathrm{W}_1, \ldots, \mathrm{W}_k$

...

An Outline for Interactive Play

Pseudocode for True instance validation

1: **for** $i = 1;\ i < \max;\ i = i + 2$ **do**
2: **Extract** $\exists$-move : M^3
3: Input $\forall$-move
4: **end for**
5: Check the Assertion

$\exists\, \mathbf{M}^1$

$\forall\, \mathbf{M}^2$

$\exists\, \mathbf{M}^3$

...

$\exists\, \mathrm{M}^d$

$\exists\, \mathrm{W}_1, \ldots, \mathrm{W}_k$

...

An Outline for Interactive Play

Pseudocode for True instance validation

1: **for** $i = 1;\ i < \max;\ i = i + 2$ **do**
2: Extract $\exists$-move
3: Input $\forall$-move
4: **end for**
5: **Check the Assertion**

$\exists\, \mathbf{M}^1$

$\forall\, \mathbf{M}^2$

$\exists\, \mathbf{M}^3$

$\ldots$

$\exists\, \mathbf{M}^d$

$\exists\, \mathbf{W}_1, \ldots, \mathbf{W}_k$

...

Static Validation using Skolem/Herbrand Functions

Interactive Play

Static Validation using Skolem/Herbrand Functions

- Input : Certificate + Assertion

Interactive Play

Static Validation using Skolem/Herbrand Functions

- Input : Certificate + Assertion
- Extract $\exists$ variable assignment with a SAT solver

Static Validation using Skolem/Herbrand Functions

- Input : Certificate + Assertion
- Extract $\exists$ variable assignment with a SAT solver
- Add assertion as an assumption to the SAT solver

Static Validation using Skolem/Herbrand Functions

- Input : Certificate + Assertion
- Extract $\exists$ variable assignment with a SAT solver
- Add assertion as an assumption to the SAT solver

Interactive Play

Static Validation using Skolem/Herbrand Functions

- Input : Certificate + Assertion
- Extract $\exists$ variable assignment with a SAT solver
- Add assertion as an assumption to the SAT solver

Dynamic Validation using QBF solvers

- Input : QBF instance + Assertion

Interactive Play

Static Validation using Skolem/Herbrand Functions

- Input : Certificate + Assertion
- Extract $\exists$ variable assignment with a SAT solver
- Add assertion as an assumption to the SAT solver

Dynamic Validation using QBF solvers

- Input : QBF instance + Assertion
- Extract $\exists$ variable assignment with a QBF solver

Interactive Play

Static Validation using Skolem/Herbrand Functions

- Input : Certificate + Assertion
- Extract $\exists$ variable assignment with a SAT solver
- Add assertion as an assumption to the SAT solver

Dynamic Validation using QBF solvers

- Input : QBF instance + Assertion
- Extract $\exists$ variable assignment with a QBF solver
- Add assertion as an assumption to the QBF solver

Example Assertions

Assertions in Hex Encodings

Example Assertions

Assertions in Hex Encodings

- Goal-Assertion (GA): Goal is reached at the end of the play.

Example Assertions

Assertions in Hex Encodings

- Goal-Assertion (GA): Goal is reached at the end of the play.
- Legal-Black-Assertion (LBA): Black does not play on black positions.

Example Assertions

Assertions in Hex Encodings

- Goal-Assertion (GA): Goal is reached at the end of the play.
- Legal-Black-Assertion (LBA): Black does not play on black positions.

Example Assertions

Assertions in Hex Encodings

- Goal-Assertion (GA): Goal is reached at the end of the play.
- Legal-Black-Assertion (LBA): Black does not play on black positions.

Expected Assertion Status for Encoding variations

Inst: / Assert:	GA	LBA
LN-Hein-12	✓	✓
SN-Hein-12	✓	✓
SN-R-Hein-12	✓	✗

Validation results from SQval

- `SQval`[1] (Scalable QBF validator) is our open source tool written in python

[1] https://github.com/irfansha/SQval

Validation results from SQval

- `SQval`[1] (Scalable QBF validator) is our open source tool written in python
- For Static validation, `DepQBF` + `QRPcert` for certificate generation

[1] https://github.com/irfansha/SQval

Validation results from SQval

- `SQval`[1] (Scalable QBF validator) is our open source tool written in python
- For Static validation, `DepQBF` + `QRPcert` for certificate generation
- For Dynamic validation, `DepQBF` solver for interactive play

[1] https://github.com/irfansha/SQval

Validation results from SQval

- `SQval`[1] (Scalable QBF validator) is our open source tool written in python
- For Static validation, `DepQBF` + `QRPcert` for certificate generation
- For Dynamic validation, `DepQBF` solver for interactive play
- Each validation run 100 iterations with random player as an opponent

[1] https://github.com/irfansha/SQval

Validation results from SQval

- `SQval`[1] (Scalable QBF validator) is our open source tool written in python
- For Static validation, `DepQBF` + `QRPcert` for certificate generation
- For Dynamic validation, `DepQBF` solver for interactive play
- Each validation run 100 iterations with random player as an opponent

[1] https://github.com/irfansha/SQval

Validation results from SQval

- `SQval`[1] (Scalable QBF validator) is our open source tool written in python
- For Static validation, `DepQBF` + `QRPcert` for certificate generation
- For Dynamic validation, `DepQBF` solver for interactive play
- Each validation run 100 iterations with random player as an opponent

	Static		Dynamic	
Instance: / Assertion:	GA	LBA	GA	LBA
LN-Hein-12	100/0	100/0	100/0	100/0
SN-Hein-12	100/0	100/0	100/0	100/0
SN-R-Hein-12	100/0	**91/9**	100/0	**84/16**

No of valid/invalid runs by SQval for Hein-12

[1] https://github.com/irfansha/SQval

Equivalence Check of Similar Encodings

Equivalence Check of Encodings with Common Winning Strategies

Definition
Let ϕ_1 and ϕ_2 be two true QBFs with common variables $\mathcal{C}$.

Equivalence Check of Encodings with Common Winning Strategies

Definition
Let ϕ_1 and ϕ_2 be two true QBFs with common variables $\mathcal{C}$.

We define ϕ_1 solution-subsumes ϕ_2 ($\phi_1 \sqsubseteq \phi_2$) iff

all winning strategies for ϕ_1 are also winning for ϕ_2.

Equivalence Check of Encodings with Common Winning Strategies

Definition
Let ϕ_1 and ϕ_2 be two true QBFs with common variables $\mathcal{C}$.
We define ϕ_1 solution-subsumes ϕ_2 ($\phi_1 \sqsubseteq \phi_2$) iff
all winning strategies for ϕ_1 are also winning for ϕ_2.

A Single Equivalence Check

- Let S_1 be a winning strategy for ϕ_1.

Equivalence Check of Encodings with Common Winning Strategies

Definition
Let ϕ_1 and ϕ_2 be two true QBFs with common variables $\mathcal{C}$.
We define ϕ_1 solution-subsumes ϕ_2 ($\phi_1 \sqsubseteq \phi_2$) iff
all winning strategies for ϕ_1 are also winning for ϕ_2.

A Single Equivalence Check

- Let S_1 be a winning strategy for ϕ_1.
- Rewrite S_1 to S_2, to avoid variable clashes in ϕ_2.

Equivalence Check of Encodings with Common Winning Strategies

Definition
Let ϕ_1 and ϕ_2 be two true QBFs with common variables $\mathcal{C}$.
We define ϕ_1 solution-subsumes ϕ_2 ($\phi_1 \sqsubseteq \phi_2$) iff
all winning strategies for ϕ_1 are also winning for ϕ_2.

A Single Equivalence Check

- Let S_1 be a winning strategy for ϕ_1.
- Rewrite S_1 to S_2, to avoid variable clashes in ϕ_2.
- Check if the new formula $\phi_2 \wedge S_2$ is also True.

Equivalence Check of Encodings with Common Winning Strategies

Definition
Let ϕ_1 and ϕ_2 be two true QBFs with common variables $\mathcal{C}$.
We define ϕ_1 solution-subsumes ϕ_2 ($\phi_1 \sqsubseteq \phi_2$) iff
all winning strategies for ϕ_1 are also winning for ϕ_2.

A Single Equivalence Check

- Let S_1 be a winning strategy for ϕ_1.
- Rewrite S_1 to S_2, to avoid variable clashes in ϕ_2.
- Check if the new formula $\phi_2 \wedge S_2$ is also True.
- If False, the encodings are not equivalent.

Results on Equivalence Check for Hein-12

Subsumption (Q1 $\sqsubseteq$ Q2) test results from SQval

Q1: / Q2:	LN	SN	SN-R
LN	T	T	T
SN	T	T	T
SN-R	**F**	**F**	T

Results on Equivalence Check for Hein-12

Subsumption (Q1 $\sqsubseteq$ Q2) test results from SQval

Q1: / Q2:	LN	SN	SN-R
LN	T	T	T
SN	T	T	T
SN-R	**F**	**F**	T

- SN-R allows overwriting black moves, whereas LN and SN does not.

Results on Equivalence Check for Hein-12

Subsumption (Q1 $\sqsubseteq$ Q2) test results from SQval

Q1: / Q2:	LN	SN	SN-R
LN	T	T	T
SN	T	T	T
SN-R	**F**	**F**	T

- SN-R allows overwriting black moves, whereas LN and SN does not.
- Each subsumption checks take a few seconds at most.

Scaling Validation

Scalable Validation with Partial Certificates

Static Validation

Dynamic Validation

Scalable Validation with Partial Certificates

Static Validation

- One time computation of Skolem/Herbrand functions

Dynamic Validation

Scalable Validation with Partial Certificates

Static Validation

- One time computation of Skolem/Herbrand functions
- Validation iterations with SAT solver

Dynamic Validation

Scalable Validation with Partial Certificates

Static Validation

- One time computation of Skolem/Herbrand functions
- Validation iterations with SAT solver
- Certifying solvers can be slow

Dynamic Validation

Scalable Validation with Partial Certificates

Static Validation

- One time computation of Skolem/Herbrand functions
- Validation iterations with SAT solver
- Certifying solvers can be slow
- Certificates can exceed GB easily

Dynamic Validation

Scalable Validation with Partial Certificates

Static Validation

- One time computation of Skolem/Herbrand functions
- Validation iterations with SAT solver
- Certifying solvers can be slow
- Certificates can exceed GB easily

Dynamic Validation

- Any QBF solver can be used

Scalable Validation with Partial Certificates

Static Validation

- One time computation of Skolem/Herbrand functions
- Validation iterations with SAT solver
- Certifying solvers can be slow
- Certificates can exceed GB easily

Dynamic Validation

- Any QBF solver can be used
- Non-certifying QBF solvers perform well

Scalable Validation with Partial Certificates

Static Validation

- One time computation of Skolem/Herbrand functions
- Validation iterations with SAT solver
- Certifying solvers can be slow
- Certificates can exceed GB easily

Dynamic Validation

- Any QBF solver can be used
- Non-certifying QBF solvers perform well
- Each validation iteration needs to solve a QBF instance

Scalable Validation with Partial Certificates

Static Validation + Dynamic Validation : Hybrid

- Using partial certificates for first k layers, usually $k = 3$ is enough

Scalable Validation with Partial Certificates

Static Validation + Dynamic Validation : Hybrid

- Using partial certificates for first k layers, usually k $= 3$ is enough
- For rest of the layers, we interactively play with a QBF solver

Scalable Validation with Partial Certificates

Static Validation + Dynamic Validation : Hybrid

- Using partial certificates for first k layers, usually k $= 3$ is enough
- For rest of the layers, we interactively play with a QBF solver
- Partial certificates are orders of magnitude more compact.

Scalable Validation with Partial Certificates

Static Validation + Dynamic Validation : Hybrid

- Using partial certificates for first k layers, usually k $= 3$ is enough
- For rest of the layers, we interactively play with a QBF solver
- Partial certificates are orders of magnitude more compact.
- Solving QBF formulas with fewer quantifiers is several orders faster.

Validating Hein-09 with Partial Certificates

A Hein-09 instance:

- open positions : 10
- moves : 9
- solving time : 1203 seconds
- certificate size : 7.8GB

Validating Hein-09 with Partial Certificates

Certificate sizes for Hein-09 instances

		Partial Certificate				
Instance	Full Certificate	k=1	k=3	k=5	k=7	k=9
LN-Hein-09	7.8G	108	1.1K	21.8K	434K	8.2M
SN-Hein-09	641.6M	96	1.1K	23.4K	437.3K	8.2M
SN-R-Hein-09	532.7M	96	1.2K	24.2K	461.1K	8.7M

Size in Bytes

Validating Goal Assertion (GA)

	static		dynamic		hybrid-L3	
Instance	PM	TT	PM	TT	PM	TT
LN-Hein-09	-	TO	64.1	1203	**1.54**	**1.6**
SN-Hein-09	20.08K	2100	1.54	9.4	**1.53**	**0.4**
SN-R-Hein-09	18.35K	1226	**1.53**	5.4	**1.53**	**0.4**

Peak Memory (PM) in MB and Time Taken (TT) in seconds

Equivalence Check with partial certificates

- We can check subsumption using partial certificates

Equivalence Check with partial certificates

- We can check subsumption using partial certificates
- Subsumption is harder than non-subsumption

Equivalence Check with partial certificates

- We can check subsumption using partial certificates
- Subsumption is harder than non-subsumption
- In case of Hein-09, we could check non-subsumption within 8 minutes.

Equivalence Check with partial certificates

- We can check subsumption using partial certificates
- Subsumption is harder than non-subsumption
- In case of Hein-09, we could check non-subsumption within 8 minutes.

Equivalence Check with partial certificates

- We can check subsumption using partial certificates
- Subsumption is harder than non-subsumption
- In case of Hein-09, we could check non-subsumption within 8 minutes.

	LN-Hein-09		SN-Hein-09		SN-R-Hein-09	
Inst:	Full	L9	Full	L9	Full	L9
LN-Hein-09	-	1.53	-	1.57	-	1.57
SN-Hein-09	63.7	1.53	63.7	1.6	63.7	1.6
SN-R-Hein-09	50.74	1.6	50.78	1.6	50.8	1.69

Peak Memory for subsumption check in GB

Conclusion

Summary

- Validation using interactive play + assertions

SQval

Conclusion

Summary

- Validation using interactive play + assertions
  - Static : using Skolem/Herbrand functions

SQval

Conclusion

Summary

- Validation using interactive play + assertions
  - Static : using Skolem/Herbrand functions
  - Dynamic : using QBF solver

SQval

Conclusion

Summary

- Validation using interactive play + assertions
  - Static : using Skolem/Herbrand functions
  - Dynamic : using QBF solver
- Equivalence checks with common winning strategies

SQval

Conclusion

Summary

- Validation using interactive play + assertions
  - Static : using Skolem/Herbrand functions
  - Dynamic : using QBF solver
- Equivalence checks with common winning strategies
- Partial certificates for scaling validation

SQval

Conclusion

Summary

- Validation using interactive play + assertions
  - Static : using Skolem/Herbrand functions
  - Dynamic : using QBF solver
- Equivalence checks with common winning strategies
- Partial certificates for scaling validation

SQval

Conclusion

Summary

- Validation using interactive play + assertions
  - Static : using Skolem/Herbrand functions
  - Dynamic : using QBF solver
- Equivalence checks with common winning strategies
- Partial certificates for scaling validation

SQval

Future Work

- Partial traces from QBF solver would help for scalable validation

Conclusion

Summary

- Validation using interactive play + assertions
  - Static : using Skolem/Herbrand functions
  - Dynamic : using QBF solver
- Equivalence checks with common winning strategies
- Partial certificates for scaling validation

SQval

Future Work

- Partial traces from QBF solver would help for scalable validation
- Allowing subsumption check beyond encodings with common variables