Model Checking Contest

Report for 2011

Fabrice Kordon - LIP6/MoVe, UPMC, France
Alban Linard - CUI/SMV, Univ. Genève, Switzerland

MCC
2011

Model Checking Contest @

Contents

Contents

Special thanks for those who helped to organize this MCC, in particular Clément Démoulins (infrastructure), Nicolas Gibelin (infrastructure and cluster), Lom Hillah (PNML), Emmanuel Paviot-Adet (models), Steve Hostettler (Properties) and Alexis Marechal (models)

MCC 2011

Objectives

When it Comes to Deal with Large and Complex Systems...

- Lots of questions are raised...
  - To verify highly concurrent systems, should we use a symmetry-based or a partial order-based model checker?
  - For models with large variable domains, should we use decision diagram-based, or a symmetry-based model checker?
  - Can we combine structural reductions techniques with partial-order ones or symmetry-based ones?
  - ...
- A large variety of model checking techniques
  - and their potential combination
- A large variety of model categories
- A challenge with large scale specifications
- A need to evaluate in the fairest way current MC implementations

The Objectives...

- MCC is intended to:
  - Exchange experience between tool programmers,
  - Imagine some association of techniques, and thus better tools
  - Stimulate development of tools
  - Provide visibility to these tools
- MCC can also be of great help for the PN community (and users):
  - Define a common set of models for benchmarks
  - Identify experimentally classes of problems (in models)
    - identify the techniques able to cope with a given class of problems...
  - Improve communication between tools (and PNML ;-))
  - Provides raw data for comparison
- This is a first edition
  - We hope more editions for an enhanced analysis and evaluation of tools

MCC 2011

Evaluation Procedure

What to be measured?

- The «enemies» of model checking
  - Memory consumption
  - CPU consumption
- «Examinations» to be processed
  - State space generation
  - Deadlock detection
  - Formula evaluation
    - Reachability
    - Verified and unverified
  - Etc... (for next editions)
    - Temporal formulae (LTL, CTL) ?
    - Stochastic analysis?
    - Time analysis?

Evaluation procedure

- Execution on a dedicated cluster
  - 22 2.4GHz bi-Pentium Xeon with 2Gbyte
  - One machine fully dedicated to tools (minimal preemption)
  - Tools must process «examination» for each models
    - Scale parameter to evaluate how far tools can run

- Run = execution of a tool for one examination on one model/scale

- A benchmark script launching all runs
  - with time confinement — 1800 sec per run
  - with memory confinement — 1.75 GByte per run
  - with both time and memory measures
  - deployment + execution on the cluster via OAR
    - OAR = resource manager for cluster developed at INRIA

Submission and evaluation procedures

- Submission
  - Step 0: read the rules to check for submission condition
  - Step 1: download the submission kit (benchmark script + PNML files)
  - Step 2: integration of the tool in the execution system
    - translate the models (or parse PNML ;-)) + parse formulas
    - wrap the tool for Unix and provide appropriate outputs
    - adapt setup.sh file
  - Step 3: use the appropriate Makefile entry to build the submission archive
  - Step 4: upload your file via a dedicated web page
- Evaluation
  - Step 0: check execution of tools on the cluster + adaptations
  - Step 1: finalize the execution environment
    - Check for confinement
    - Check for the metrics capture
    - Extract results
  - Step 2: run all submissions (with final values for confinement)
  - Step 3: analyze results

Submission and evaluation procedures

- Submission
  - Step 0: read the rules to check for submission condition
  - Step 1: download the submission kit (benchmark script + PNML files)
  - Step 2: integration of the tool in the execution system
    - translate the models (or parse PNML ;-)) + parse formulas
    - wrap the tool for Unix and provide appropriate outputs
    - adapt setup.sh file
  - Step 3: use the appropriate Makefile entry to build the submission archive
  - Step 4: upload your file via a dedicated web page
- Evaluation
  - Step 1: finalize the execution environment on the cluster + adaptations
  - Step 2: run all submissions (with final values for confinement)
  - Step 3: analyze results

Memory measure was sort of a nightmare. It is not perfect yet.

However, all tools were evaluated the same way

Submission and evaluation procedures

- Submission
  - Step 0: read the rules to check for submission condition
  - Step 1: download the submission kit (benchmark script + PNML files)
  - Step 2: integration of the tool in the execution system
    - translate the models (or parse PNML ;-)) + parse formulas
    - wrap the tool for Unix and provide appropriate outputs
    - adapt setup.sh file
  - Step 3: use the appropriate Makefile entry to build the submission archive
  - Step 4: upload your file via a dedicated web page
- Evaluation
  - Step 1: finalize the execution environment
  - Step 2: run all submissions (with final values for confinement)
  - Step 3: analyze results

Memory measure was sort of a nightmare. It is not perfect yet.

However, all tools were evaluated the same way

Many thanks to the tool developers and their nice reactivity for solving troubles as well as for answering questions

MCC 2011

The Models

Presentation of the Models

- P/T Nets
  - FMS (Flexible Manufacturing System)
    - Bench from SMART
  - Kanban
    - Bench from SMART
  - MAPK (Mitogen-activated protein kinase kaskade)
    - Bench from Cottbus
- Colored Nets
  - Peterson
    - G.L. Peterson's mutual exclusion algorithm
  - Philosophers
    - Classical illustration for deadlocks
  - SharedMemory
    - Bench from GreatSPN (colored version) presented in PNPM'1989
  - TokenRing
    - Proposed by E.W. Dijkstra in CACM (1974)

Models Characteristics

	FMS	Kanban	MAPK	Peterson	Philosophers	SharedMemory	TokenRing
P/T	✓	✓	✓				
Safe				✓	✓	✓	✓
Colored				✓	✓	✓	✓
Cartesian product of color types				✓		✓	✓
Non equal guards				✓		✓	✓
Broadcast function				✓	✓	✓	
Succ & pred functions				✓	✓		✓

Models Characteristics

Next MCC
better coverage of
model characteristics

Next MCC
More models

	FMS	Kanban	MAPK	Peterson	Philosophers	SharedMemory	TokenRing
P/T	✓	✓	✓				
Safe				✓	✓	✓	✓
Colored				✓	✓	✓	✓
Cartesian product of color types				✓		✓	✓
Non equal guards				✓		✓	✓
Broadcast function				✓	✓	✓	
Succ & pred functions				✓	✓		✓

MEC 2011

Participating Tools

The Submissions

Tool Name	Team	Institution	Country	Contact Name
ACTIVITY-LOCAL	TIK	ETHZ	Switzerland	K. Lampka
AlPiNA	CUI/SMV	Univ. Geneva	Switzerland	D. Buchs
Crocodile	LIP6/MoVe	UPMC	France	M. Colange
ITS-Tools	LIP6/MoVe	UPMC	France	Y. Thierry-Mieg
LoLA	Team Rostock	Univ. Rostock	Germany	N. Lohmann and K. Wolf
PNXDD	LIP6/MoVe	UPMC	France	E. Paviot-Adet
PeTe	Stud. Group d402b	Univ. Aalborg	Denmark	J. Finnemann Jensen
Sara	Team Rostock	Univ. Rostock	Germany	H. Wimmel and K. Wolf
YASPA	TIK	ETHZ	Switzerland	K. Lampka
helena	LIPN/LCR	Univ. Paris 13	France	S. Evangelista

Supported techniques

Tool Tool	RG (Reachability Graph)	DL (Deadlock Detection)	F (Formula Evaluation)
ACTIVITY-LOCAL	EXPLICIT DECISION_DIAGRAMS OTHERS		
AlPiNA	DECISION_DIAGRAMS	DECISION_DIAGRAMS	
Crocodile	SYMMETRIES DECISION_DIAGRAMS		SYMMETRIES DECISION_DIAGRAMS
ITS-Tools	DECISION_DIAGRAMS SYMMETRIES (opt)	DECISION_DIAGRAMS SYMMETRIES (opt)	DECISION_DIAGRAMS SYMMETRIES (opt)
LoLA			EXPLICIT PARTIAL_ORDERS STATE_COMPRESSION
PNXDD	DECISION_DIAGRAMS		
PeTe			EXPLICIT ABSTRACTIONS OTHERS
Sara			ABSTRACTIONS PARTIAL_ORDERS OTHERS
YASPA	DECISION_DIAGRAMS		
helena	EXPLICIT	EXPLICIT ABSTRACTIONS PARTIAL_ORDERS	

Models & examinations Processed by Tools

	FMS	Kanban	MAPK	Peterson	Philosophers	SharedMemory	TokenRing
ACTIVITY-LOCAL	RG	RG					
AlPiNA	DL RG	DL RG	DL RG	DL RG	DL RG	DL RG	DL RG
Crocodile	F RG	F RG	F RG		F RG	F RG	
ITS-Tools	DL F RG	DL F RG	DL F RG		DL F RG	DL F RG	DL F RG
LoLA	F	F	FOK		F		
PNXDD	RG	RG	RG	RG	RG	RG	RG
PeTe	F	F	F				
Sara	F	F	F				
YASPA	RG	RG	RG (8)				
helena	DL RG	DL RG		DL RG	DL RG	DL RG	DL RG

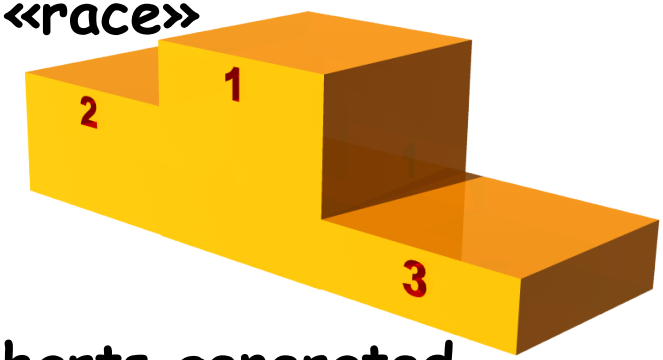
- No formula evaluation for Peterson
- Less than 50% of submitted tools handling colored nets

MCC 2011

Analysis of the Results

How to perform the analysis

- No interest in a «race»



- More than 175 charts generated
  - Synthesis to be published (TOPNOC report or tech report or web)
  - Raw data will be available on-line or as an annex to the report
  - It is annoying to show them all...

- Identification (partial) of some «surprises» discovered when test were processed
  - How tools scale up
    - P/T and colored
  - Some observations on time and memory consumption
  - Feed back with tools' characteristics

How Did Tools Scale Up (P/T Models)

red
- max scale for this model

no value
- no instance completed

		A-LOCAL	AlPiNA	Crocodile	ITS-Tools	LoLA	PNXDD	PeTe	Sara	YASPA	helena
FMS	RG	20	10	2	100		**500**			100	2
	DL		10		50						**500**
	FOK			2	200	**500**			**500**		
	FNOK			2	50	20		**500**	**500**		
Kanban	RG	20	10		200		500			50	
	DL		10		200						10
	FOK				200	200			**1000**		
	FNOK				200	10		**1000**	**1000**		
MAPK	RG		8		160		160			8	
	DL		8		160						
	FOK				160	**320**			80		
	FNOK				160	20		**320**	**320**		

Why Did Tools Failed (P/T Models)

		A-LOCAL	AlPiNA	Crocodile	ITS-Tools	LoLA	PNXDD	PeTe	Sara	YASPA	helena
FMS	RG	EDNF	EDNF	MOVF	MOVF		EDNF			MOVF	MOVF
	DL		EDNF		MOVF						
	FOK			MOVF	EDNF			MOVF			
	FNOK			MOVF	MOVF	SOVF					
Kanban	RG	EDNF	EDNF	MOVF	MOVF		MOVF			???	MOVF
	DL		EDNF		MOVF						MOVF
	FOK			MOVF	MOVF	SOVF		MOVF			
	FNOK			MOVF	MOVF	SOVF					
MAPK	RG		EDNF	MOVF	MOVF		EDNF				
	DL		EDNF		MOVF						
	FOK			MOVF	MOVF			MOVF	MOVF		
	FNOK			MOVF	MOVF	SOVF					

- EDNF = execution did not finished
- SOVF = stack overflow
- MOVF = memory overflow
- ??? = cannot determine failure

How Did Tools Scale Up (Colored Models)

red
max scale for this model

green
max scale reached for this model

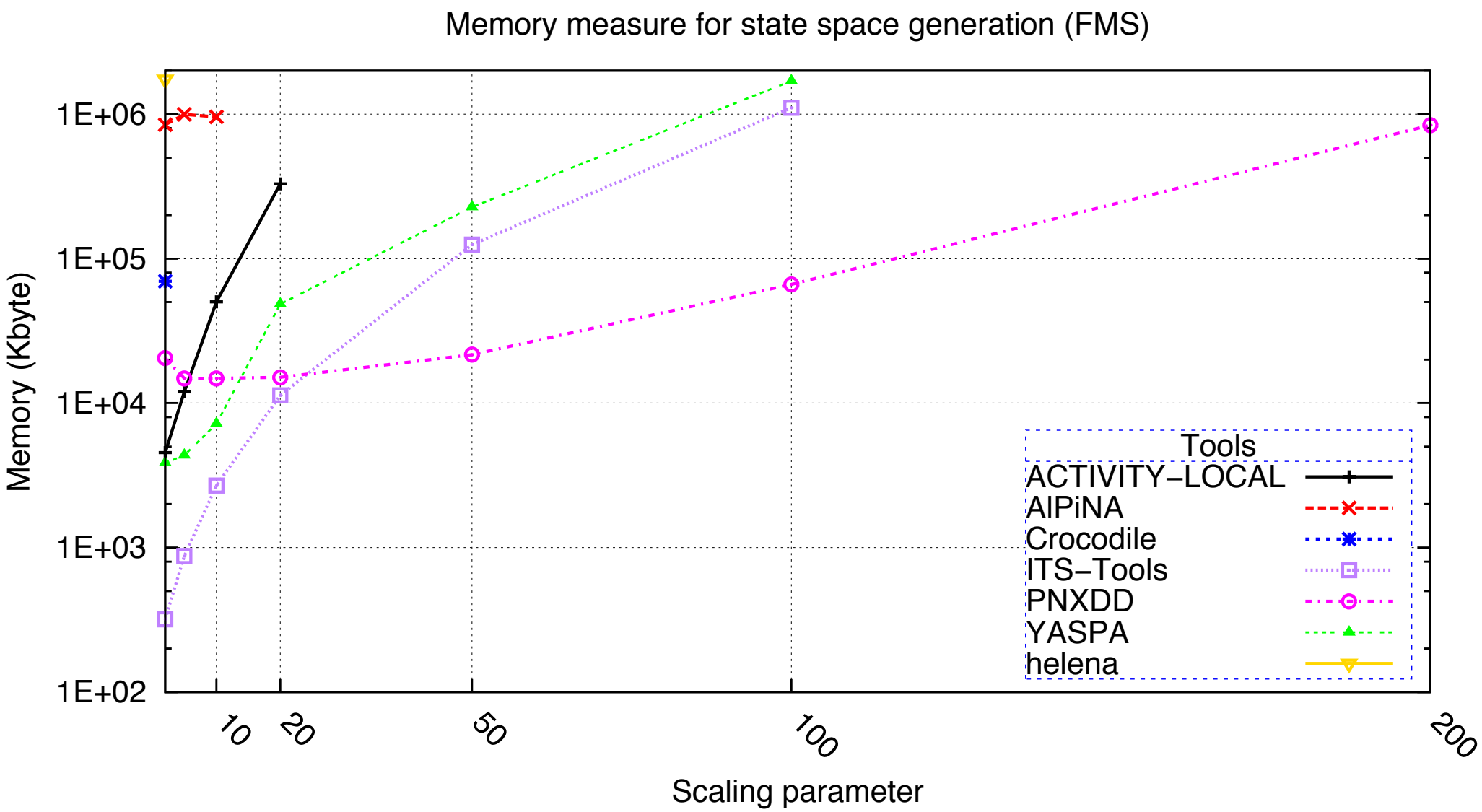
		A-LOCAL	AlPiNA	Crocodile	ITS-Tools	LoLA	PNXDD	PeTe	Sara	YASPA	helena
Peterson	RG		3				**5**				2
	DL		3								3
	FOK										
	FNOK										
Philosophers	RG		500	10	**100000**		1000				10
	DL		100		**100000**						10
	FOK			5	5000	10					
	FNOK			5	5000	10					
SharedMemory	RG		20	20	**50000**		100				10
	DL		20		**50000**						10
	FOK			10	20000						
	FNOK			10	20000						
TokenRing	RG		10		**50**		15				10
	DL		10		**50**						10
	FOK				40						
	FNOK				**50**						

Why Did Tools Failed (Colored Models)

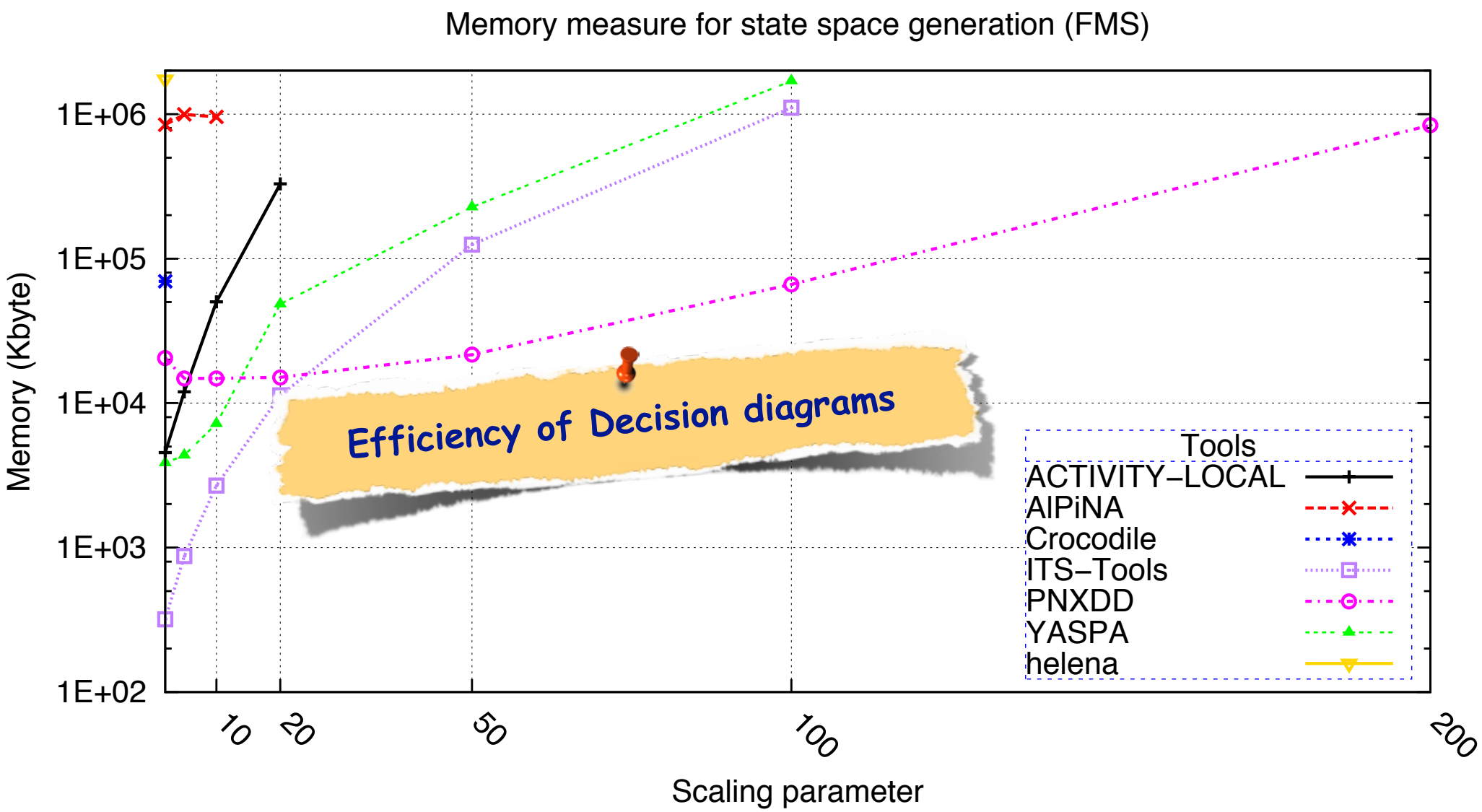
		A-LOCAL	AlPiNA	Crocodile	ITS-Tools	LoLA	PNXDD	PeTe	Sara	YASPA	helena
Peterson	RG		EDNF				???				EDNF
	DL		EDNF								EDNF
	FOK										
	FNOK										
Philosophers	RG		EDNF	EDNF			SOVF				???
	DL		EDNF								EDNF
	FOK			EDNF	MOVF	SOVF					
	FNOK			EDNF	MOVF	SOVF					
SharedMemory	RG		MOVF	MOVF			EDNF				EDNF
	DL		MOVF								EDNF
	FOK			MOVF	MOVF						
	FNOK			MOVF	MOVF						
TokenRing	RG		MOVF		MOVF		EDNF				EDNF
	DL		MOVF		MOVF						EDNF
	FOK				MOVF						
	FNOK				MOVF						

- EDNF = execution did not finished
- SOVF = stack overflow
- MOVF = memory overflow
- ??? = cannot determine failure

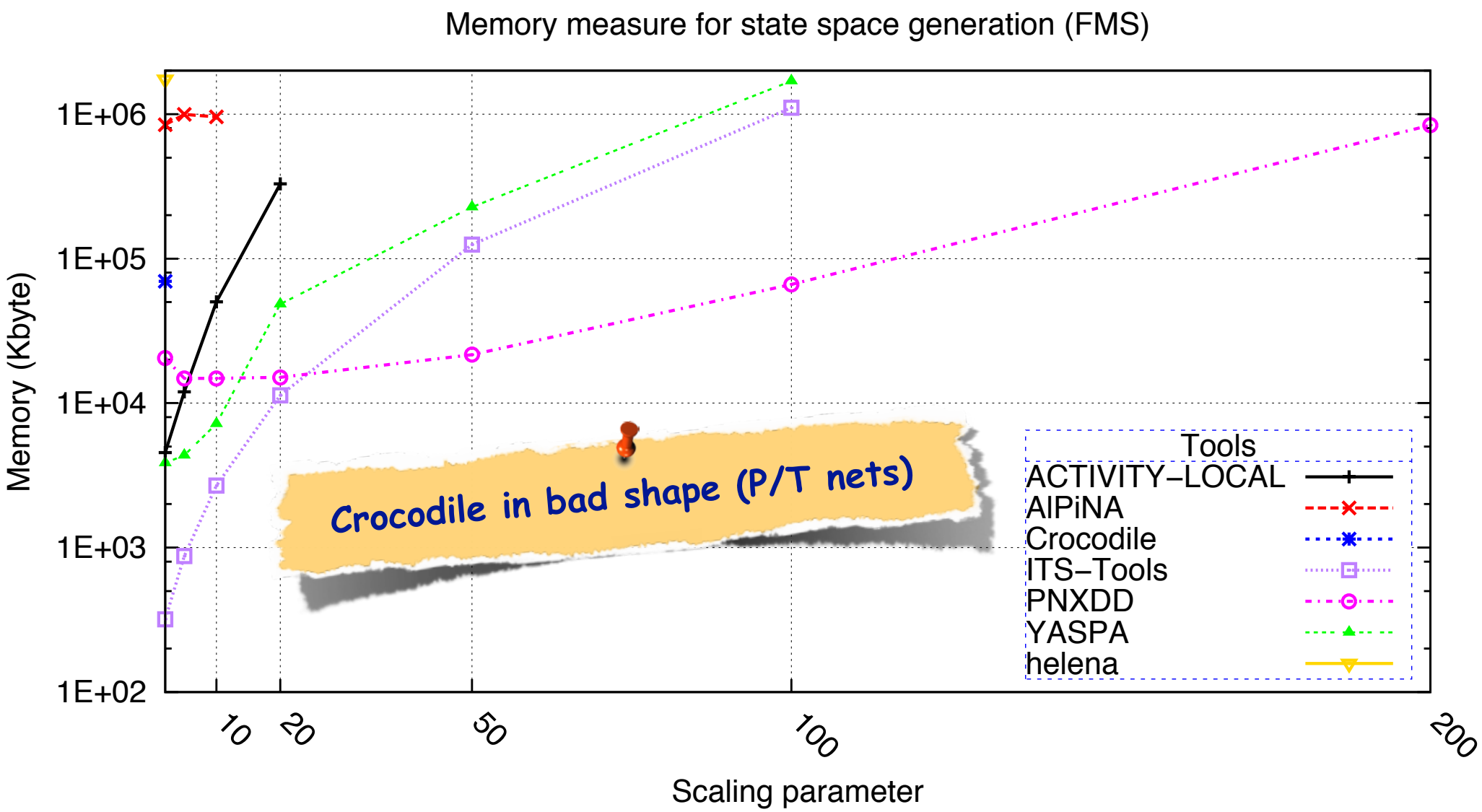
FMS, State Space Generation



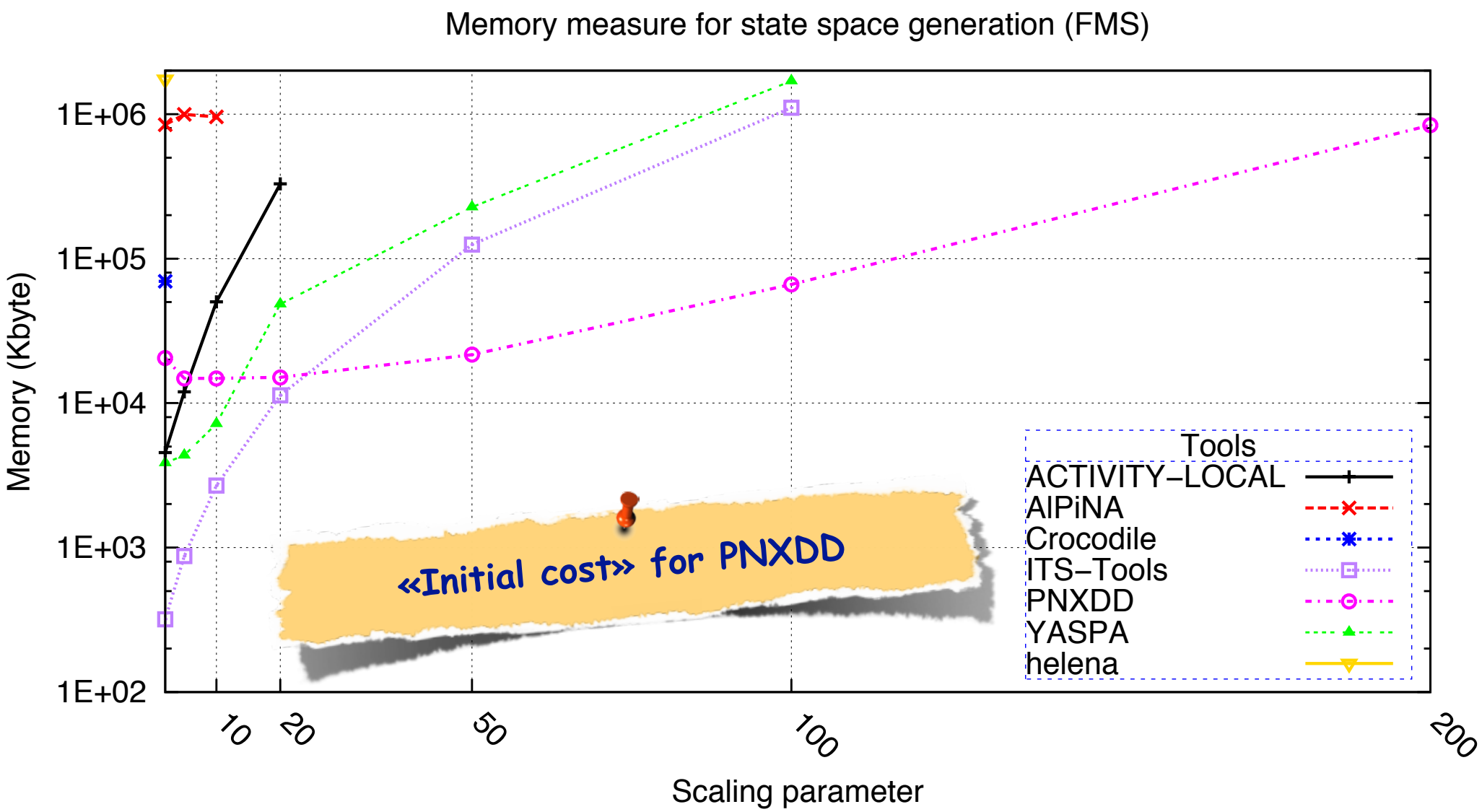
FMS, State Space Generation



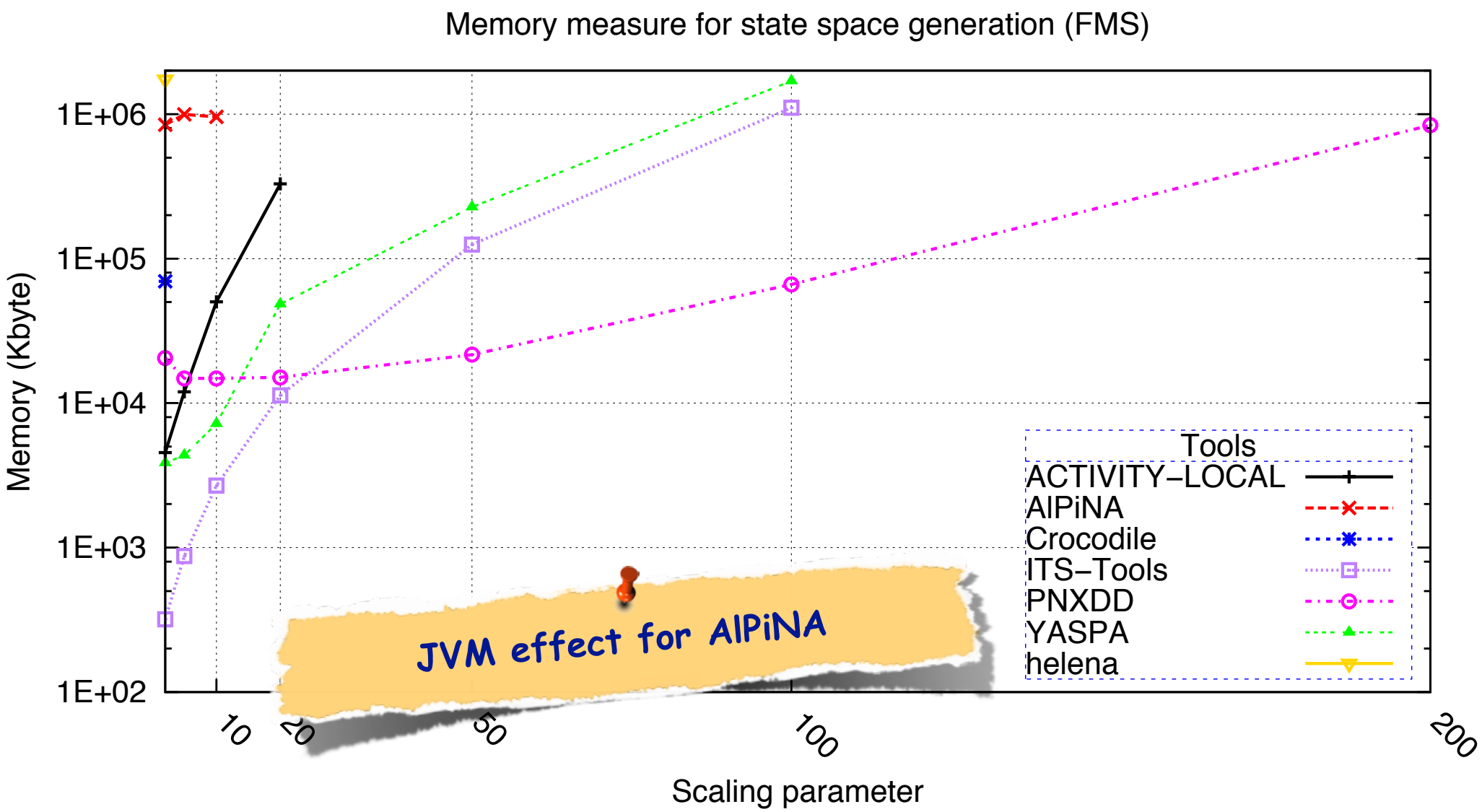
FMS, State Space Generation



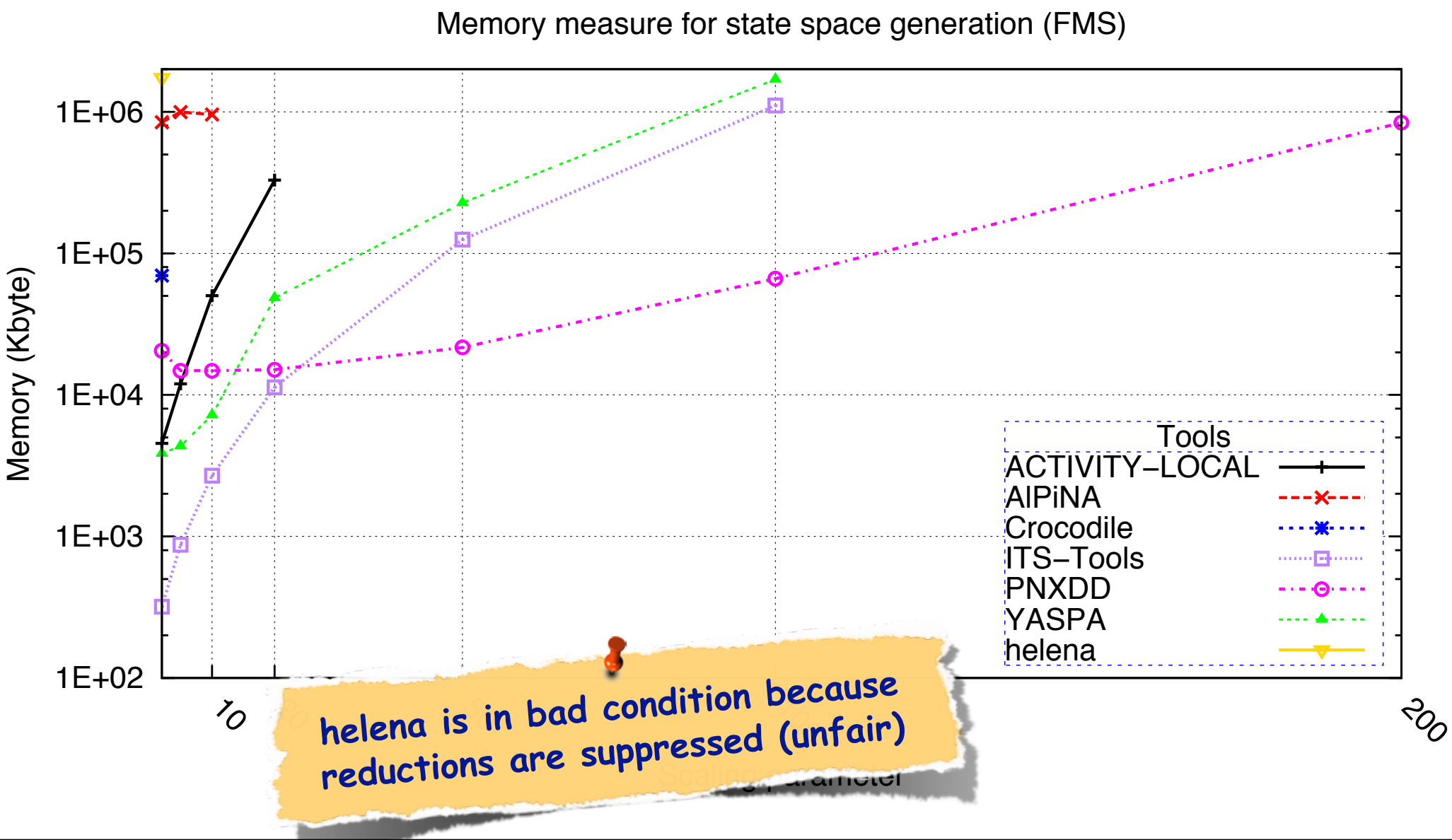
FMS, State Space Generation



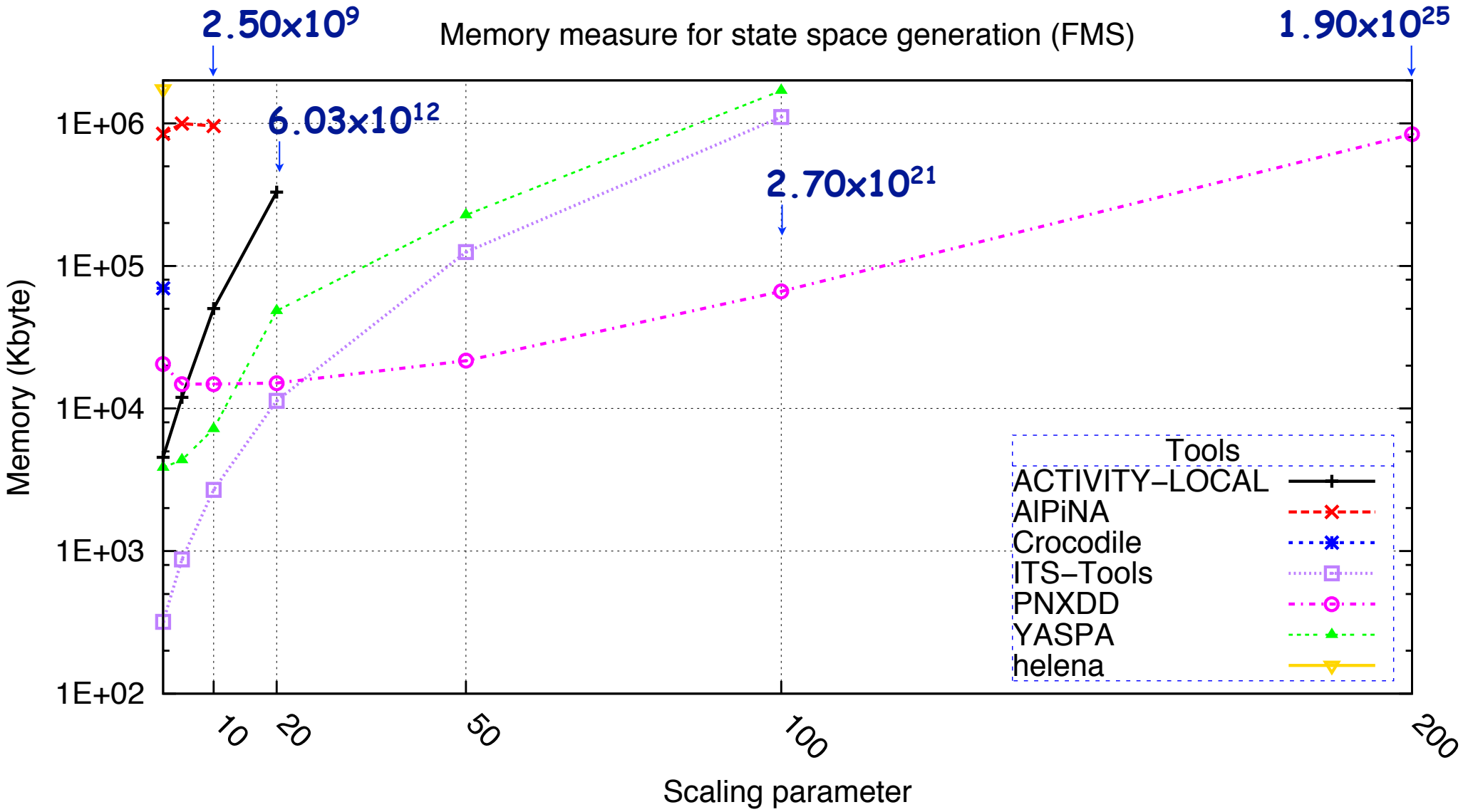
FMS, State Space Generation



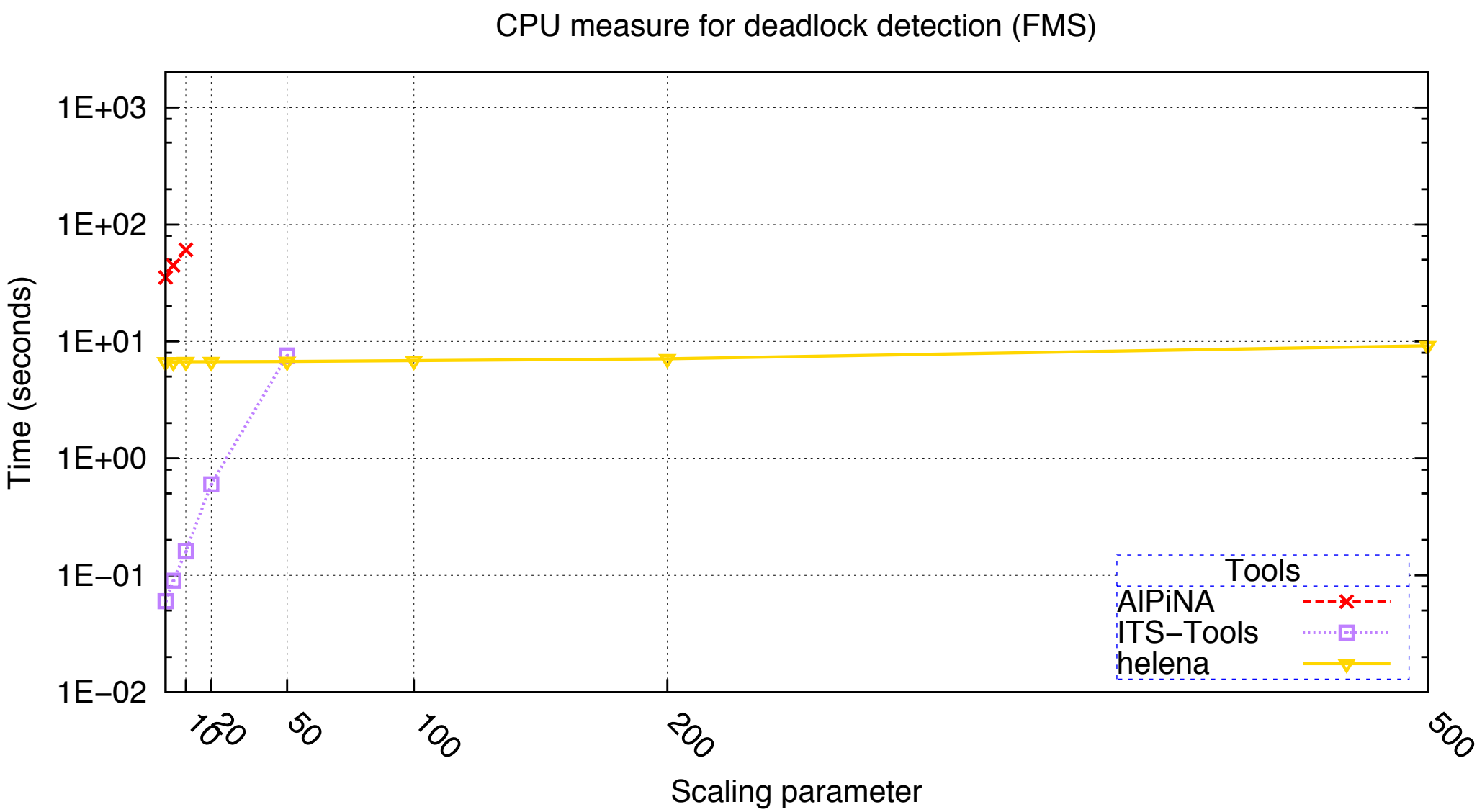
FMS, State Space Generation



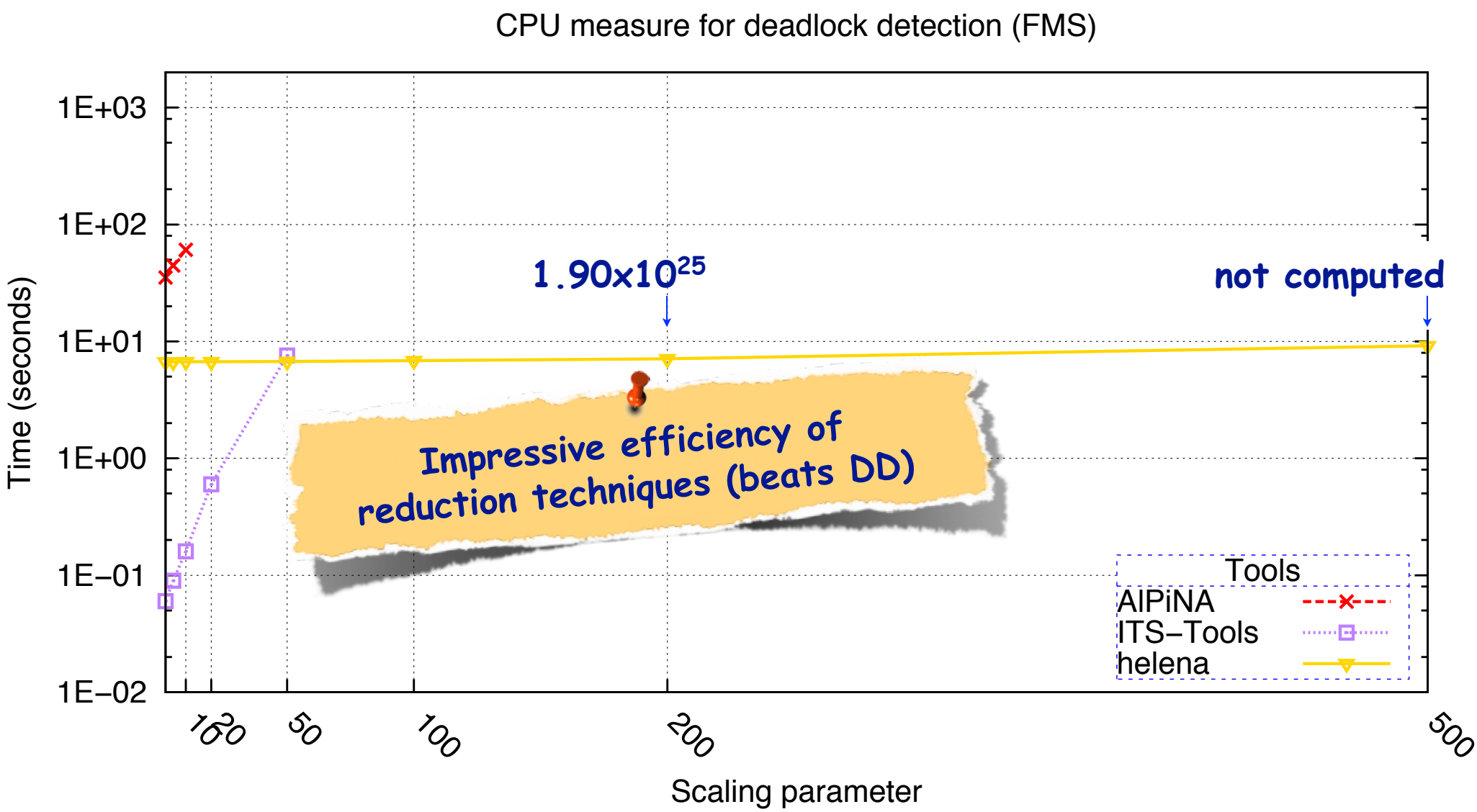
FMS, State Space Generation



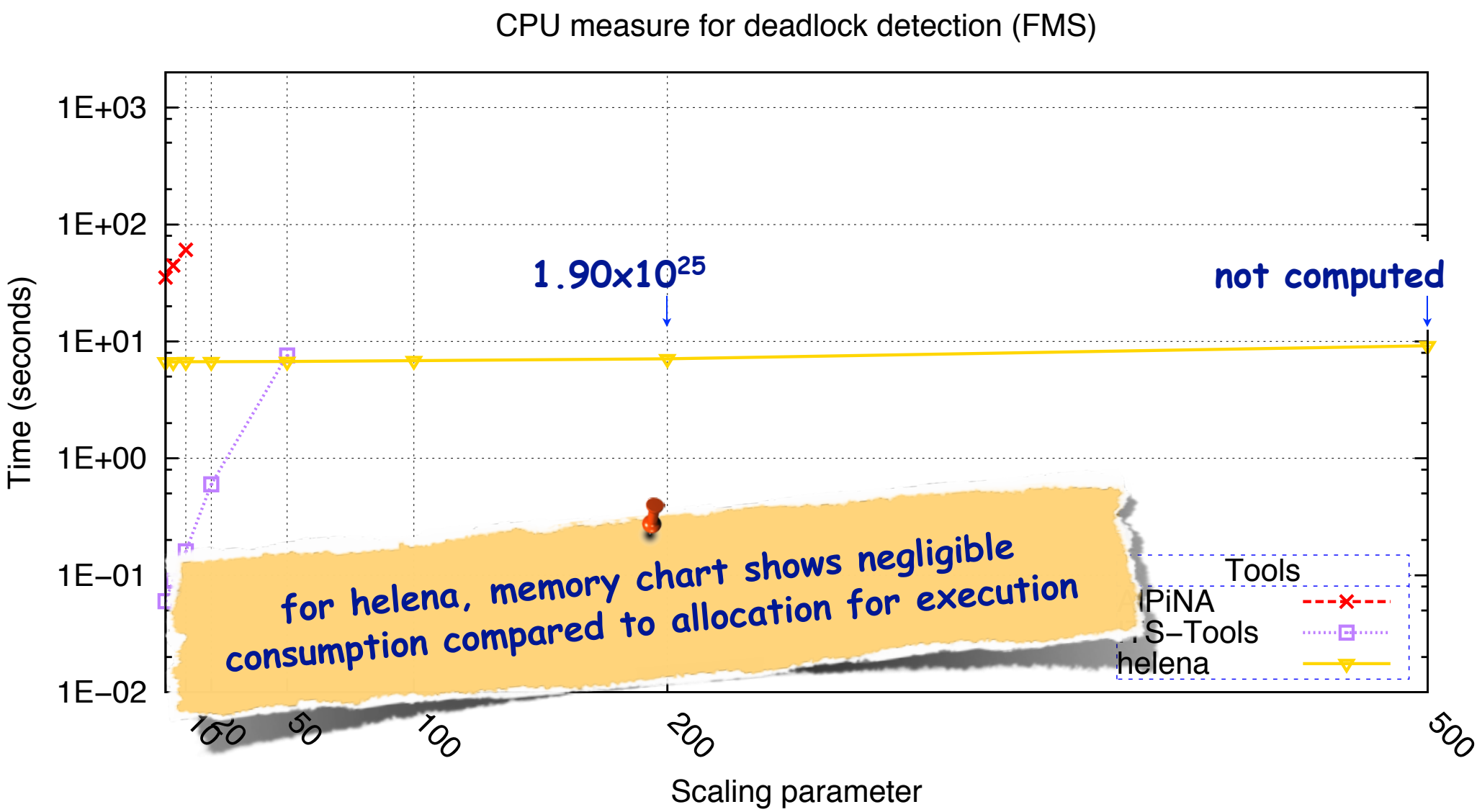
FMS, Deadlock Detection



FMS, Deadlock Detection

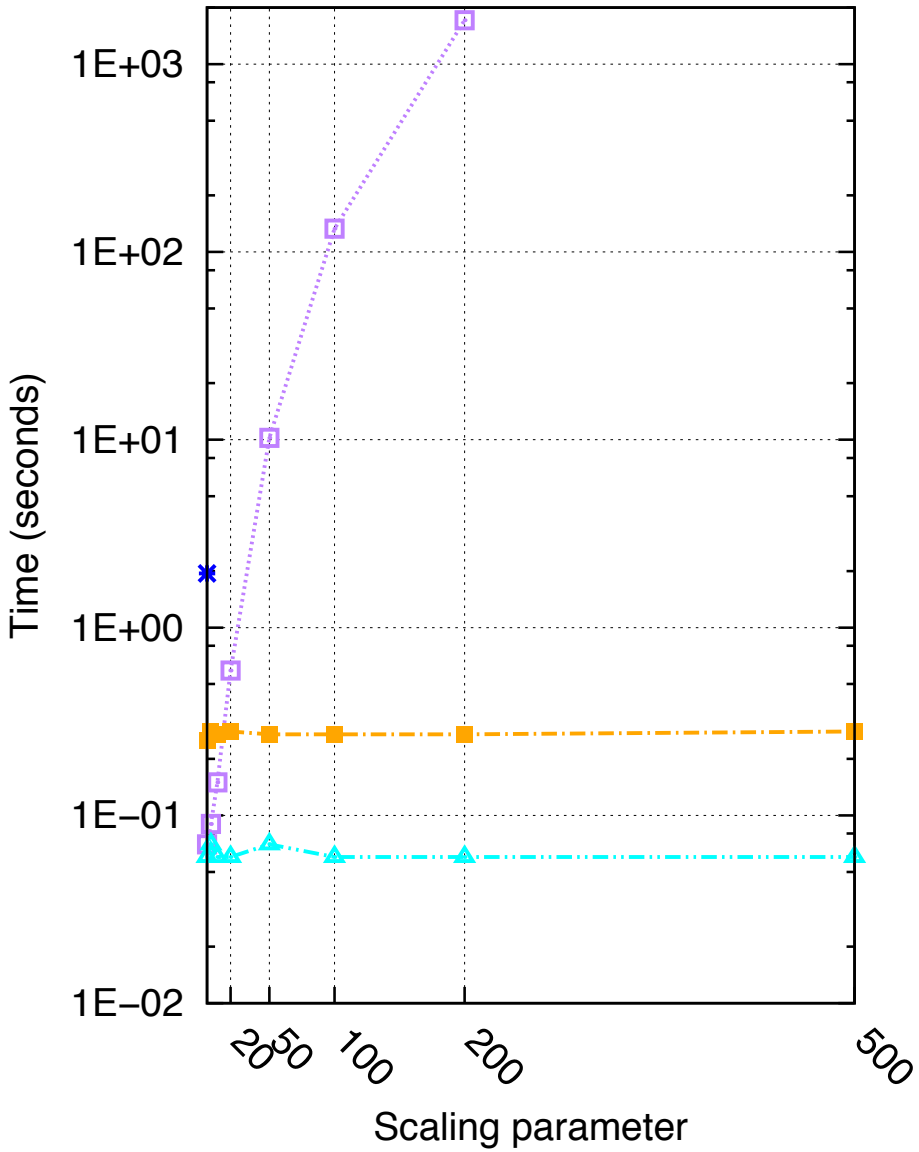


FMS, Deadlock Detection

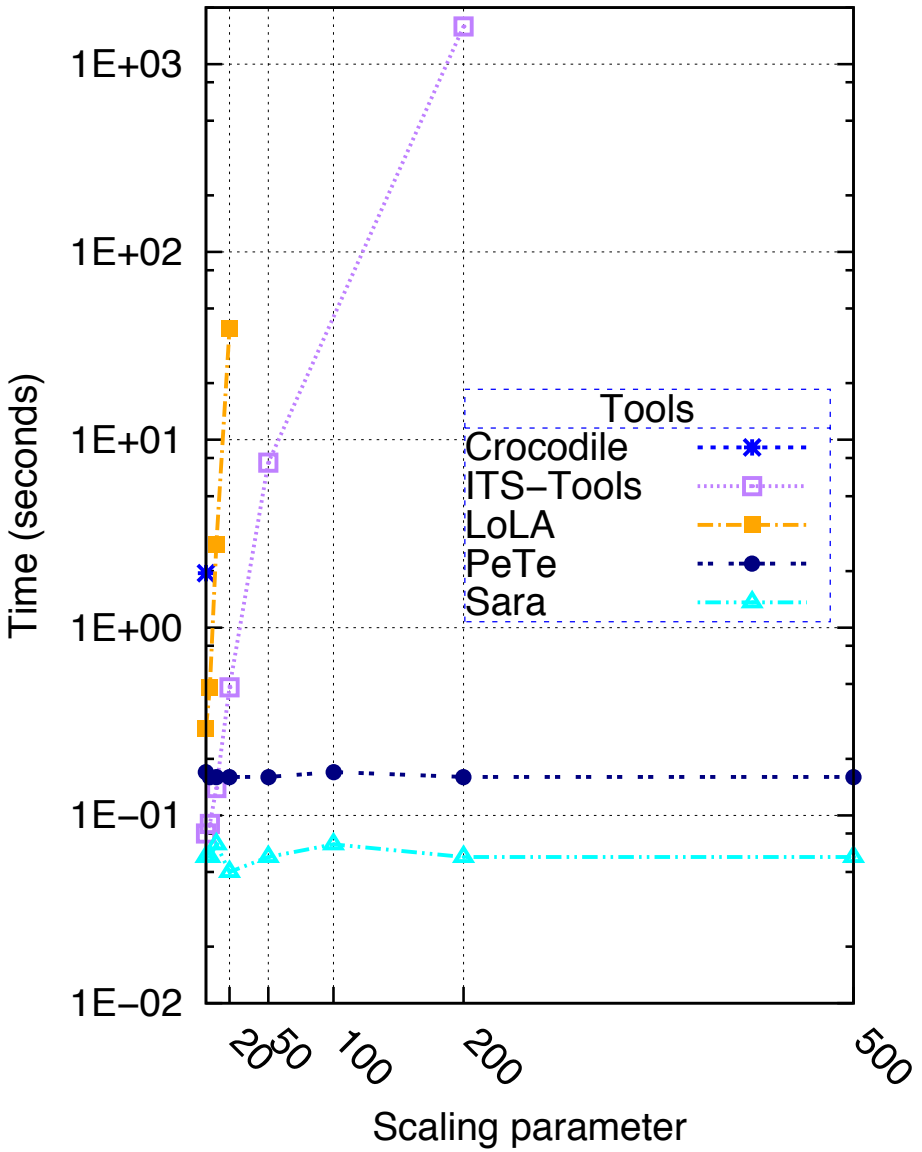


FMS, Reachability Analysis

CPU measure for the evaluation of verified formula (FMS)

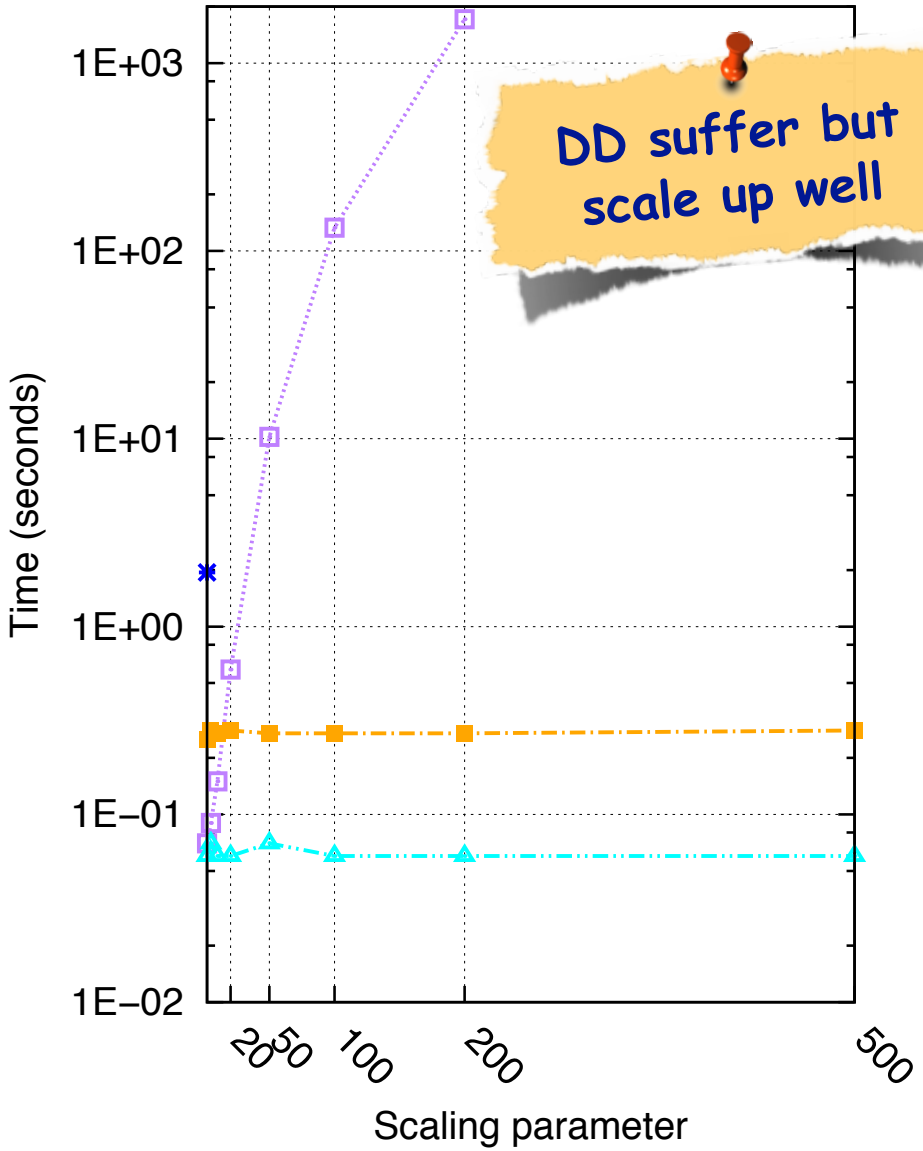


CPU measure for the evaluation of unverified formula (FMS)

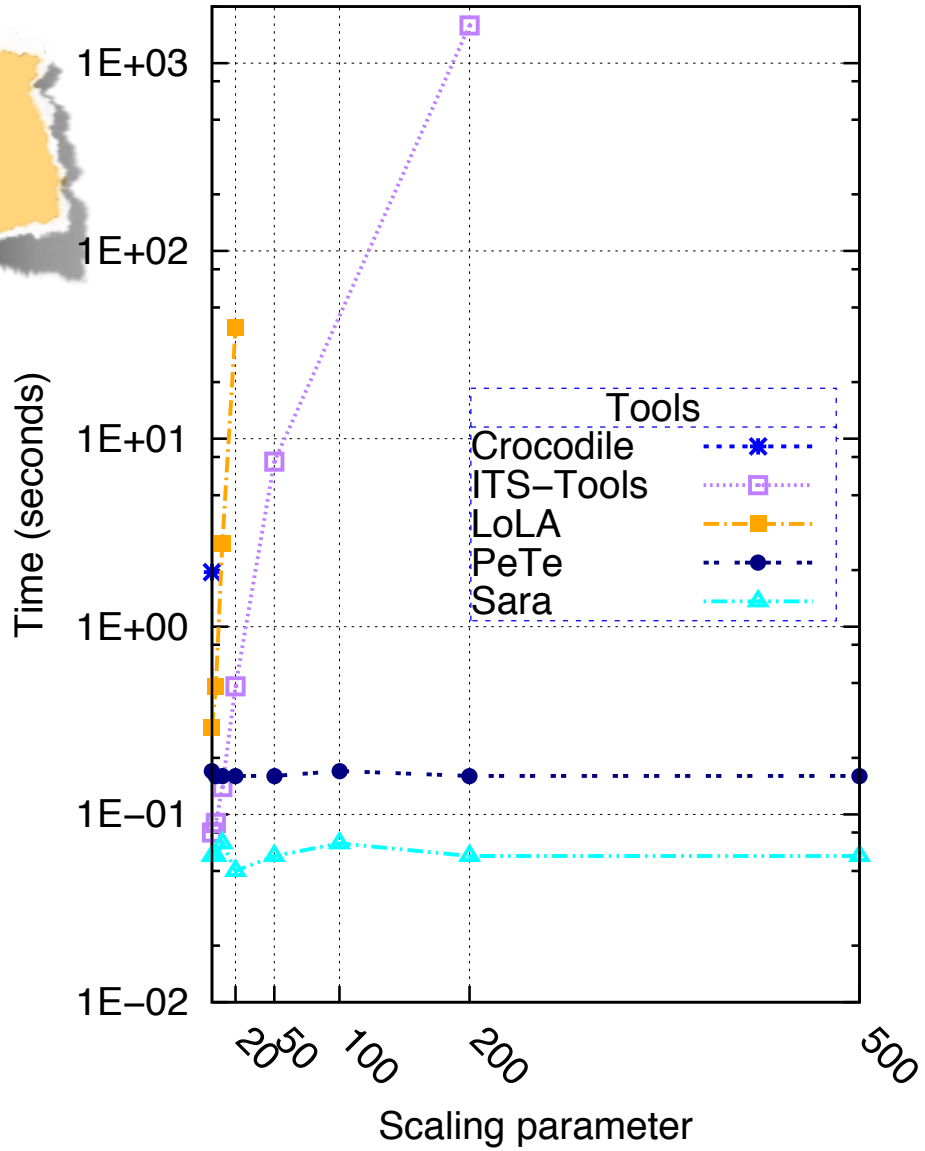


FMS, Reachability Analysis

CPU measure for the evaluation of verified formula (FMS)

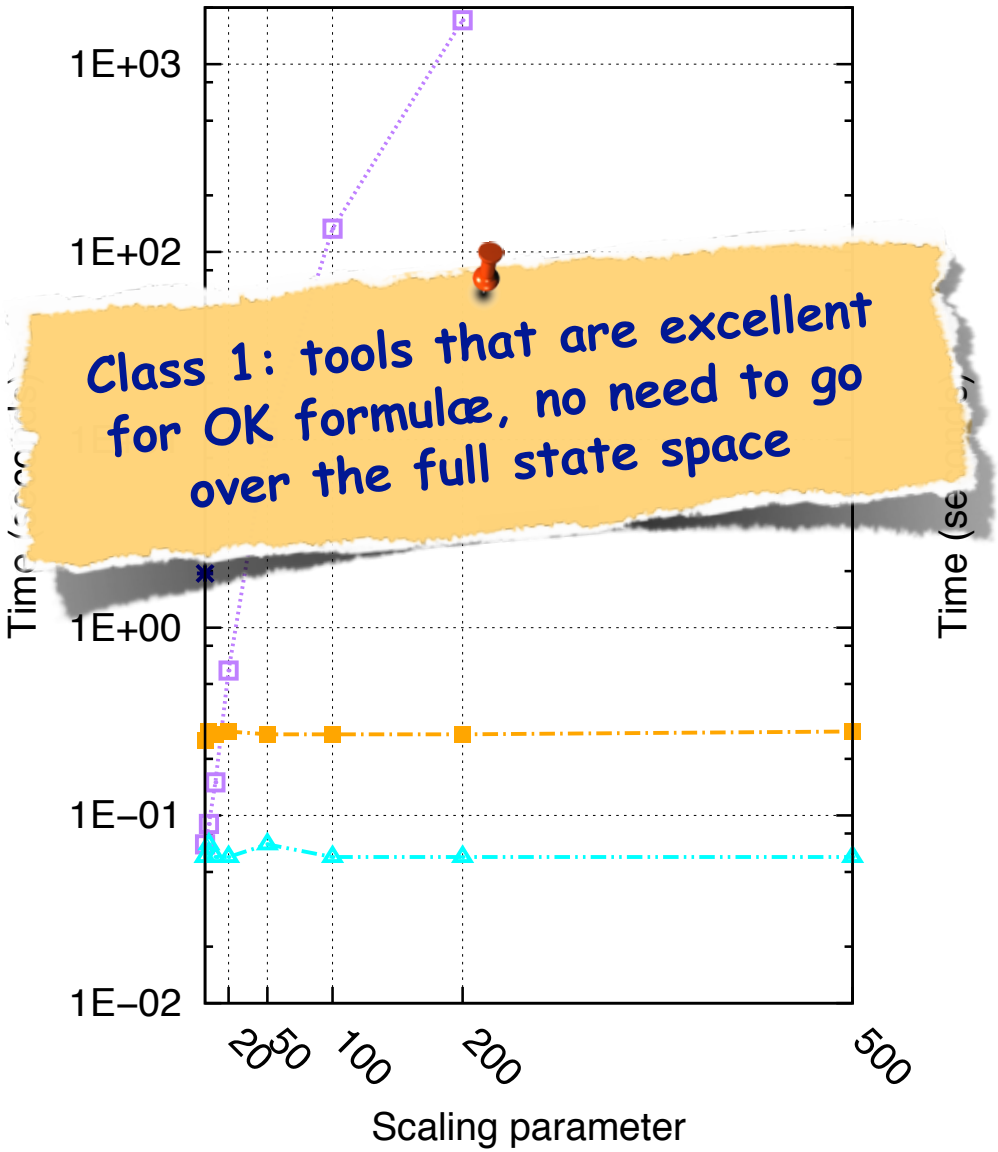


CPU measure for the evaluation of unverified formula (FMS)



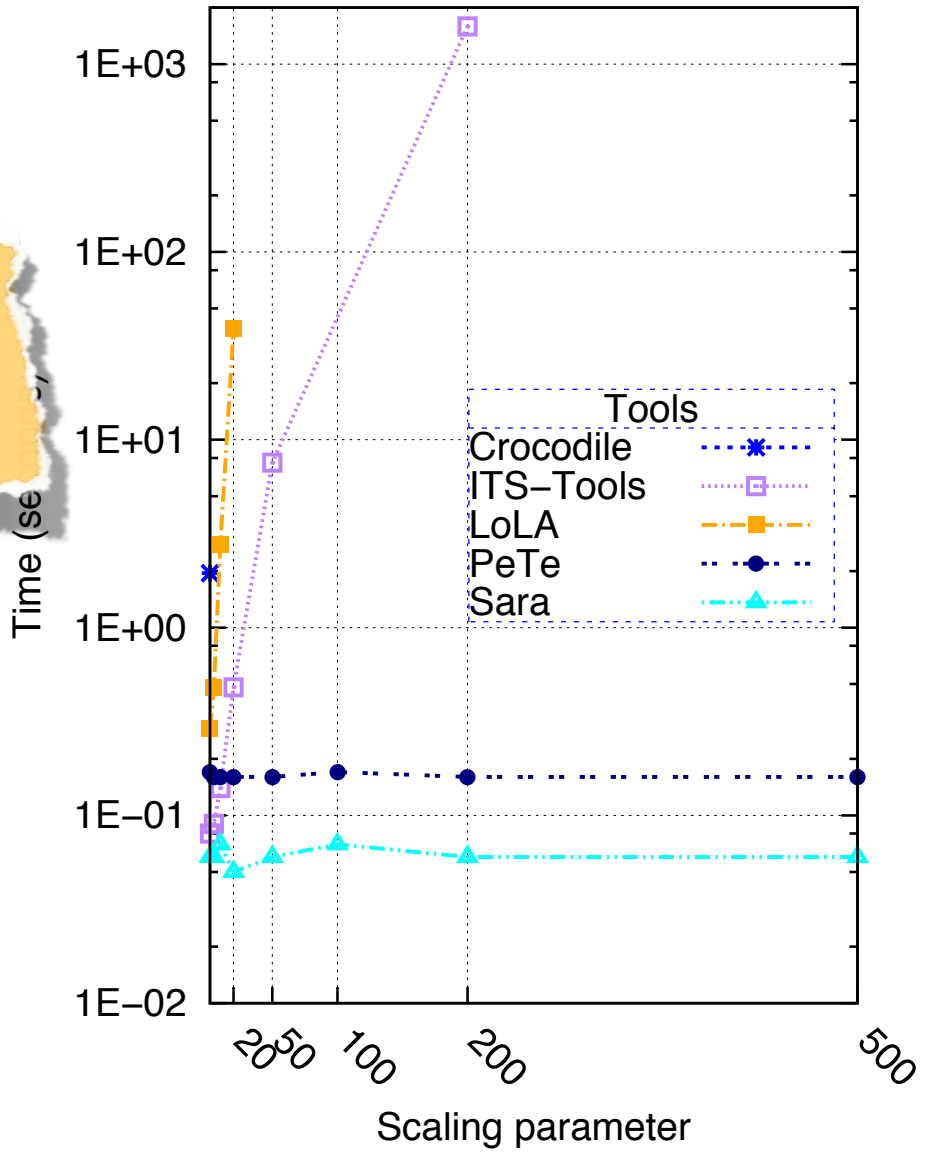
FMS, Reachability Analysis

CPU measure for the evaluation of verified formula (FMS)



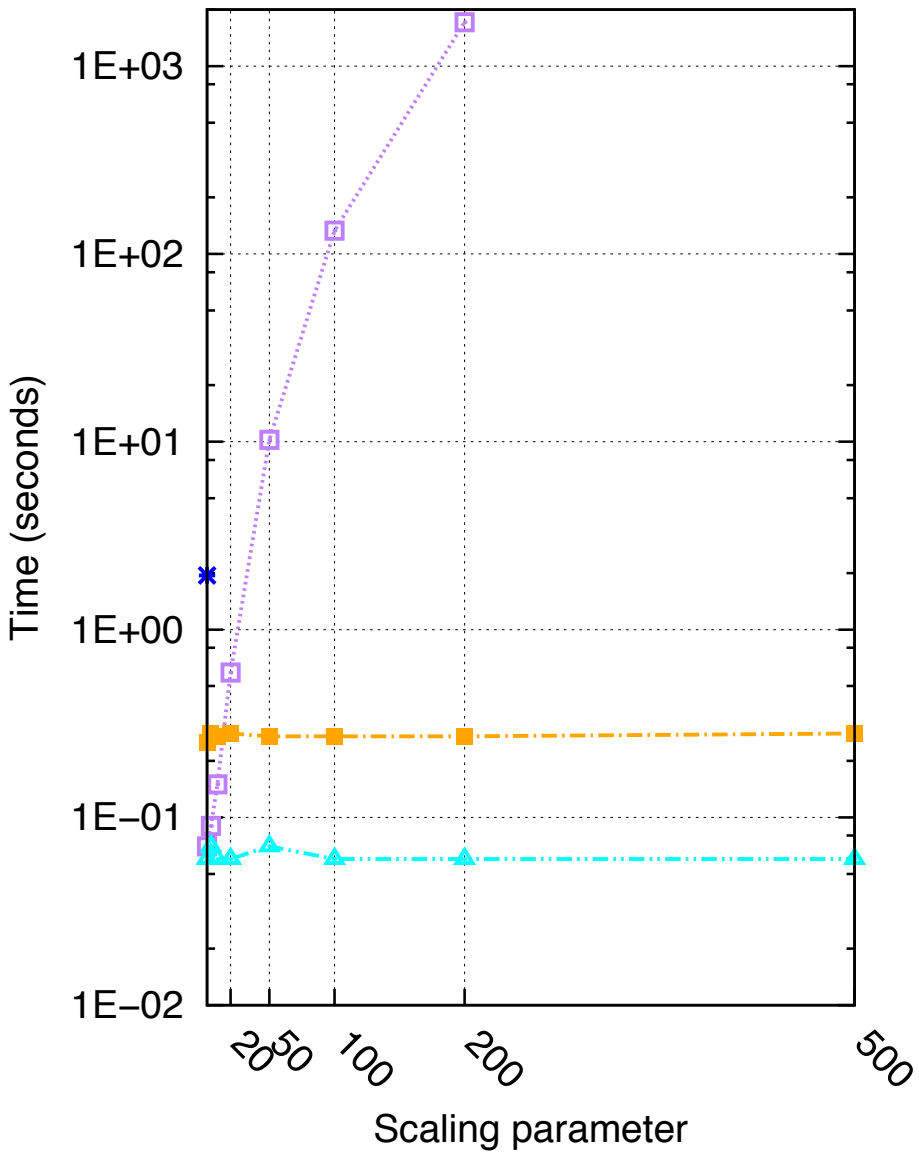
Class 1: tools that are excellent for OK formulæ, no need to go over the full state space

CPU measure for the evaluation of unverified formula (FMS)

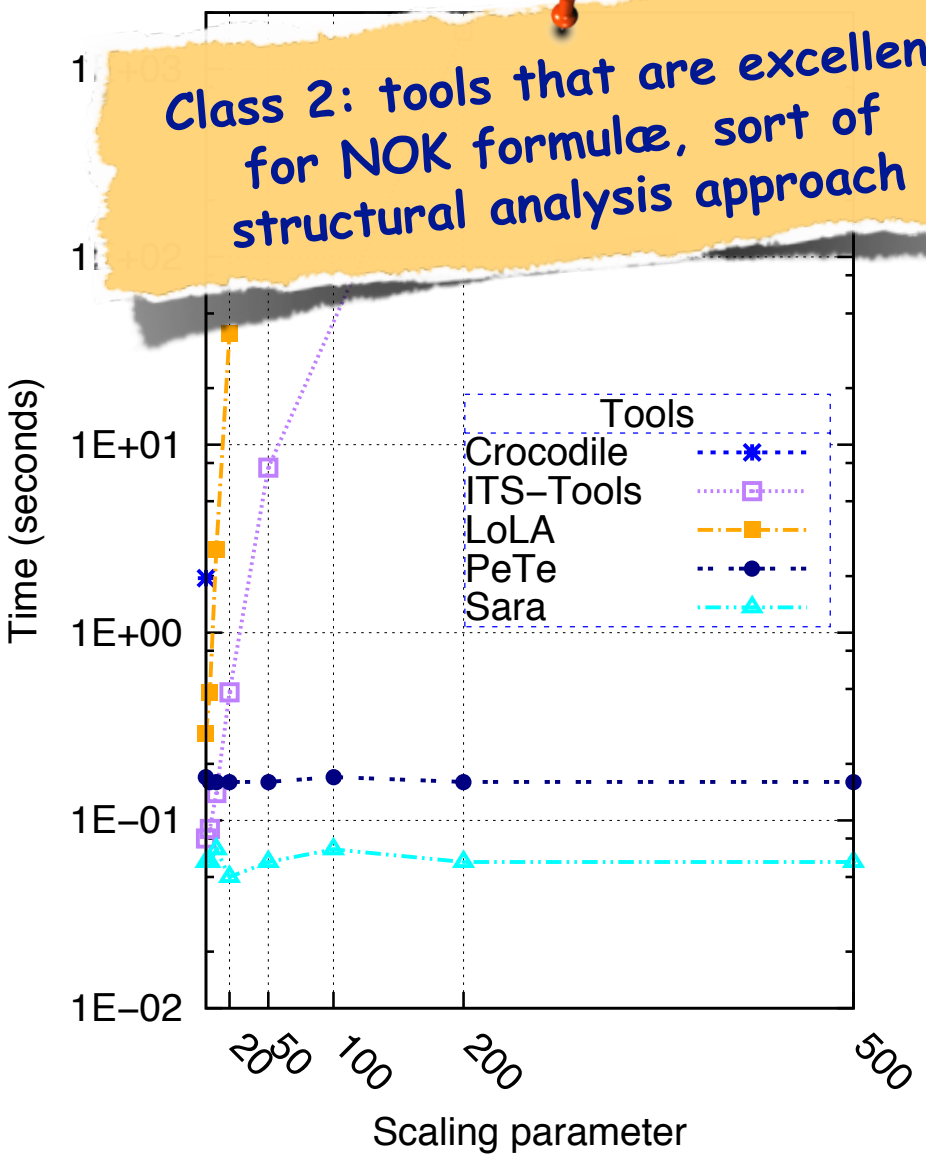


FMS, Reachability Analysis

CPU measure for the evaluation of verified formula (FMS)

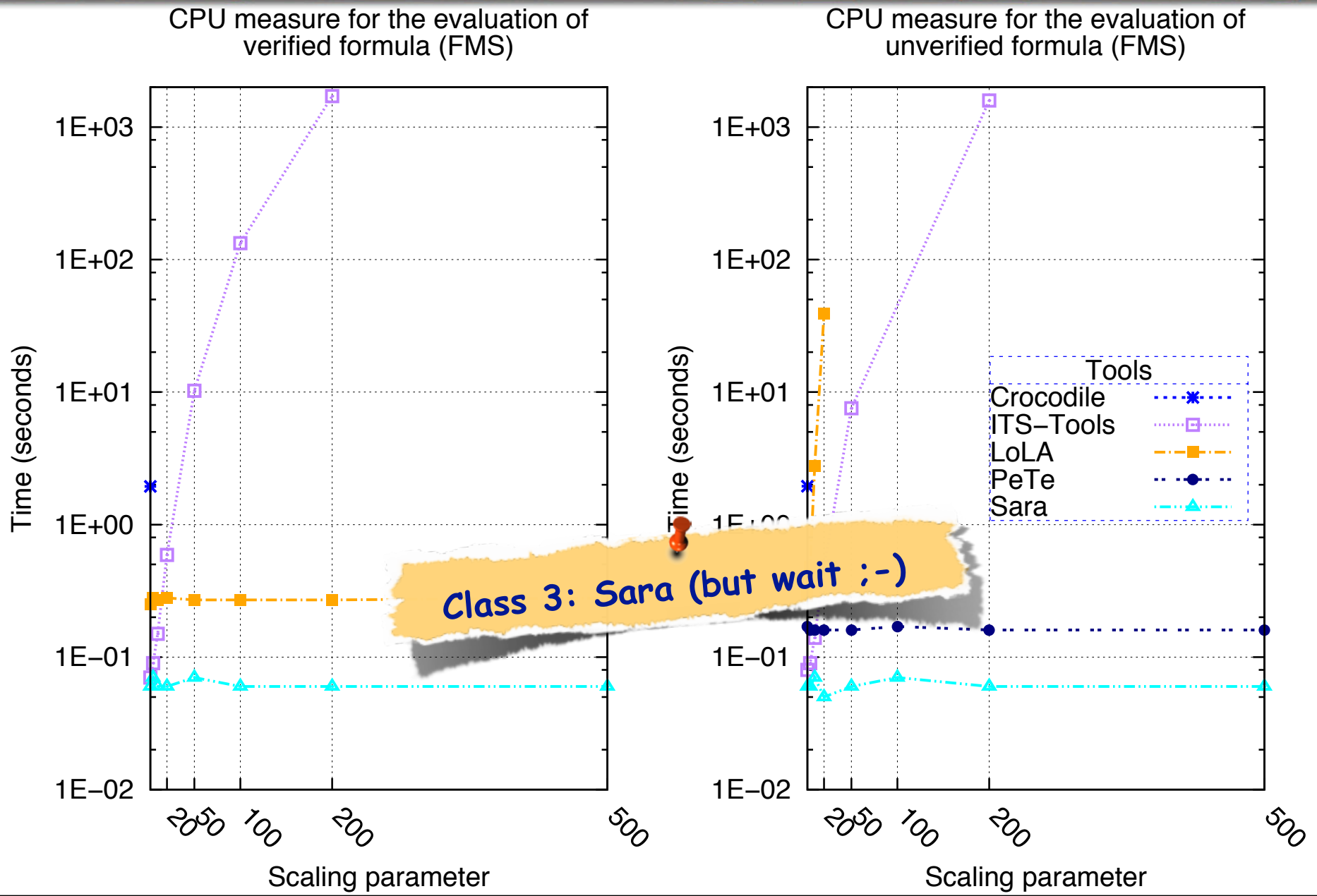


CPU measure for the evaluation of unverified formula (FMS)



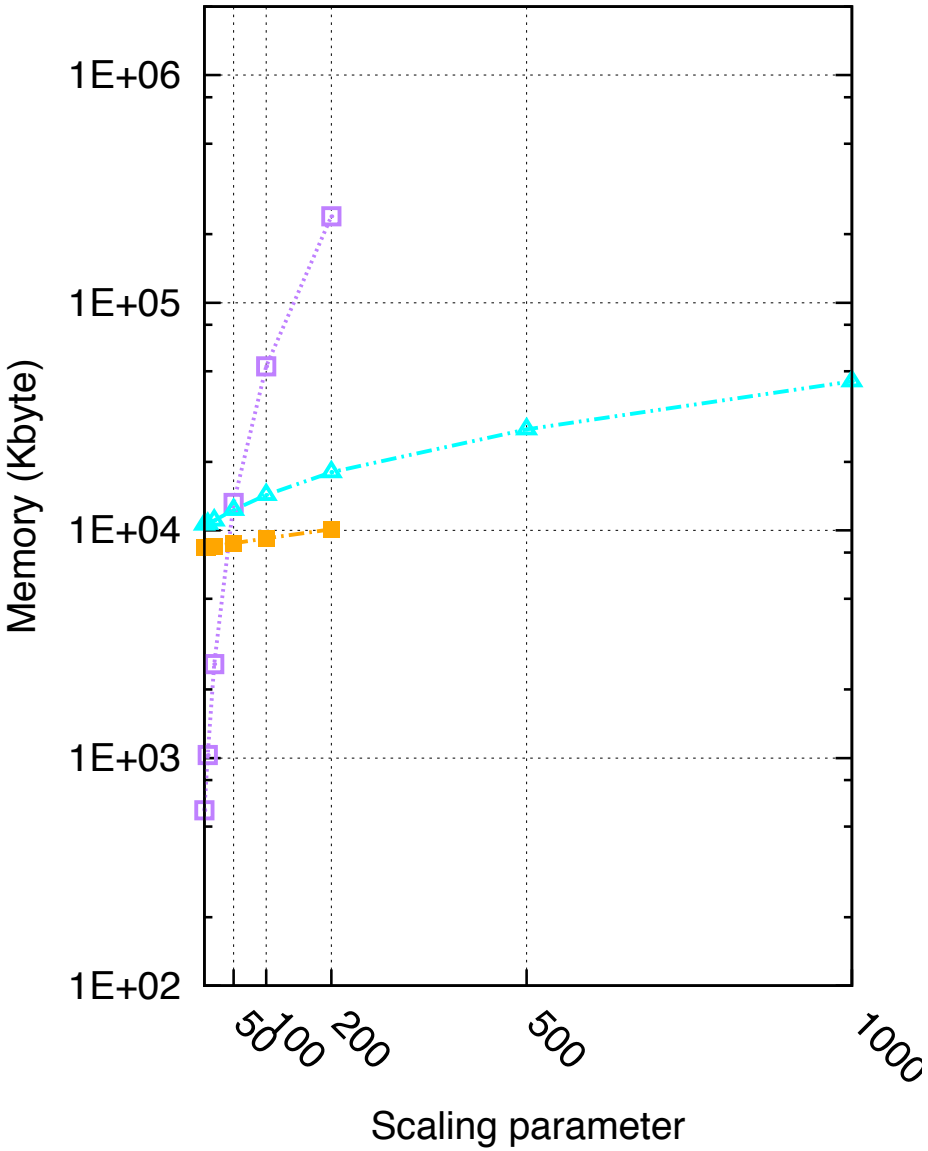
Class 2: tools that are excellent for NOK formulæ, sort of structural analysis approach

FMS, Reachability Analysis

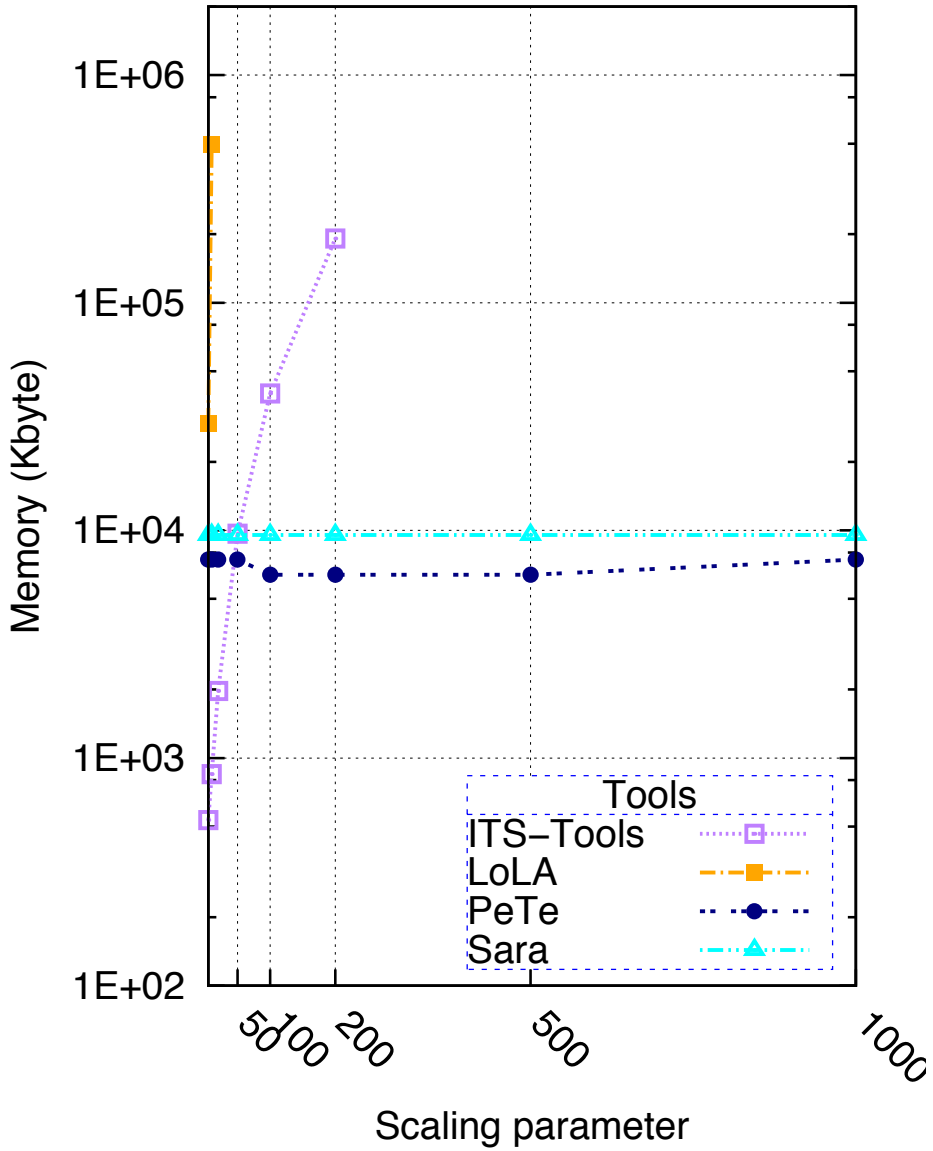


Kanban, Reachability Analysis

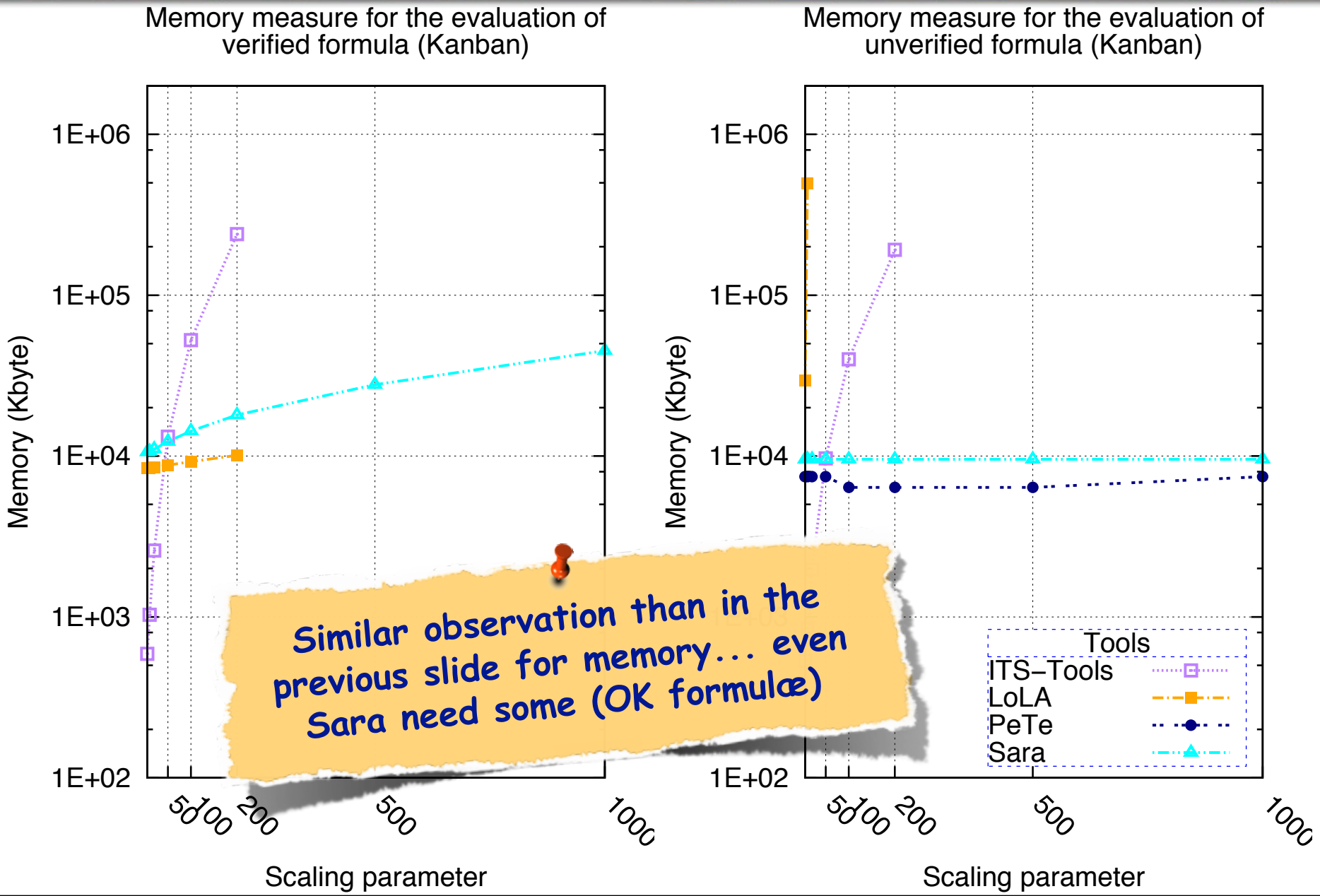
Memory measure for the evaluation of verified formula (Kanban)



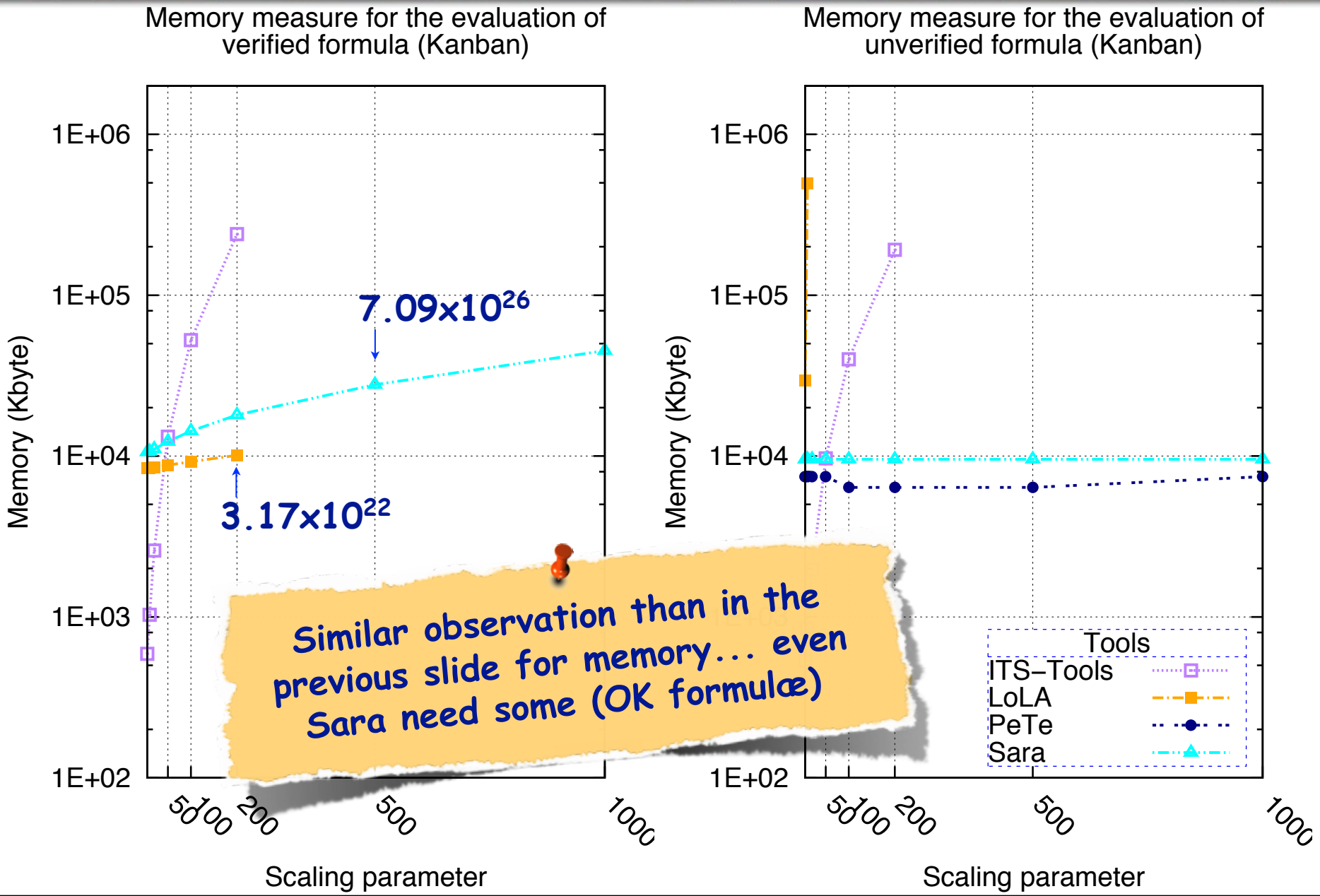
Memory measure for the evaluation of unverified formula (Kanban)



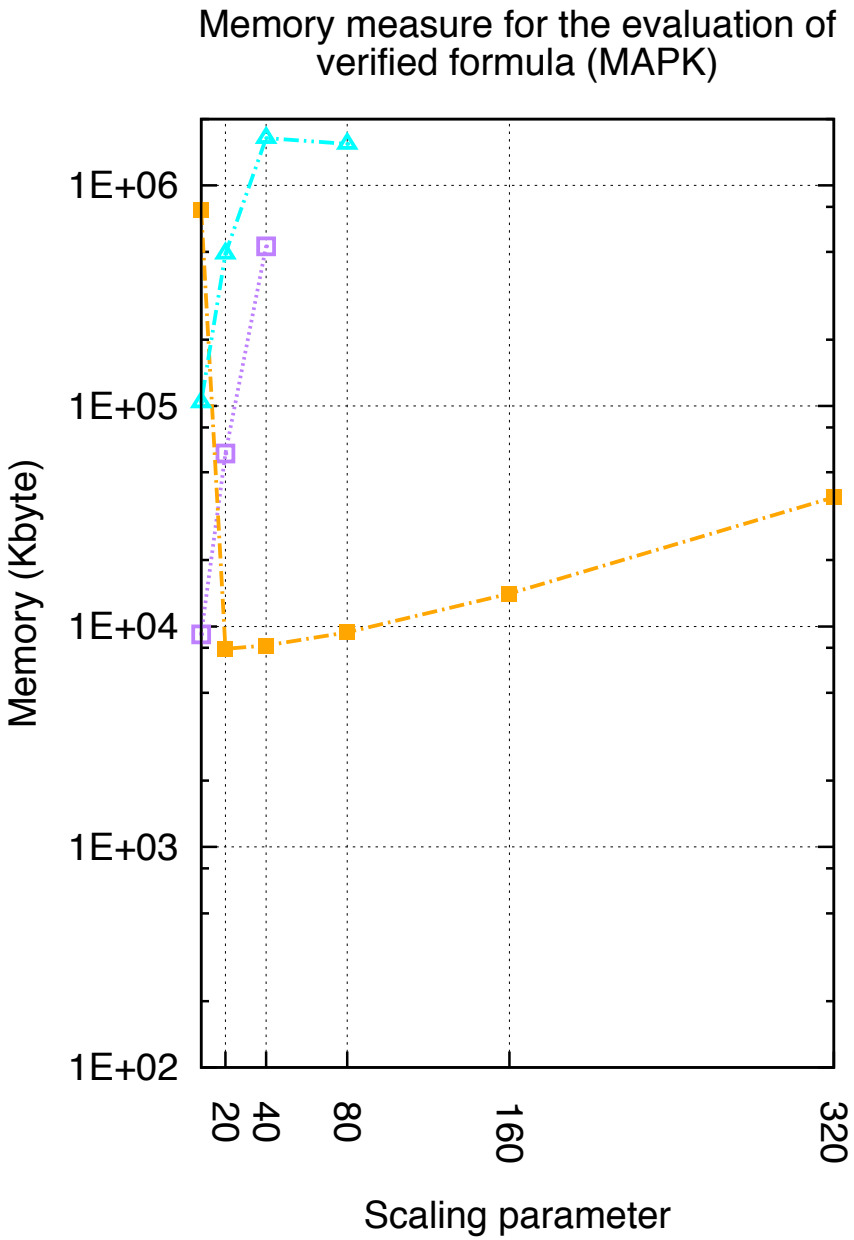
Kanban, Reachability Analysis

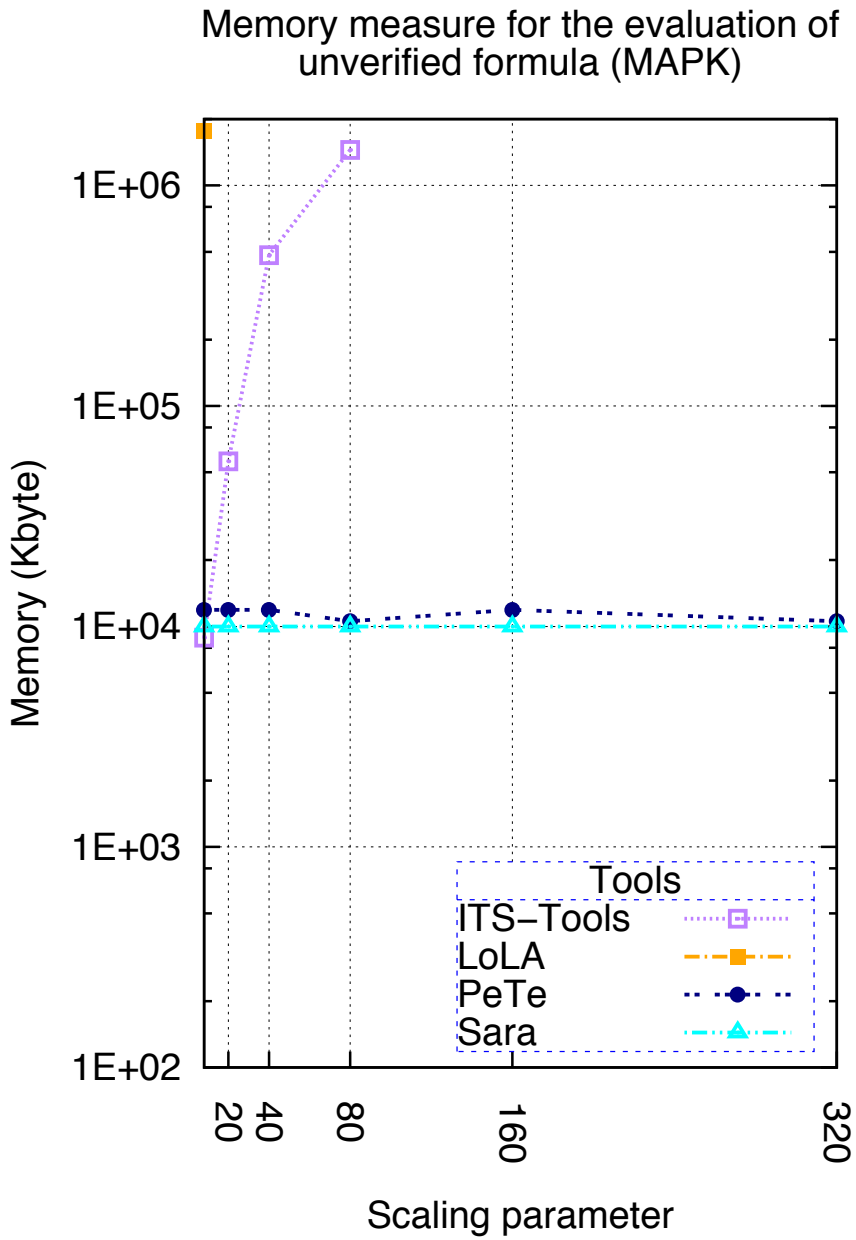


Kanban, Reachability Analysis

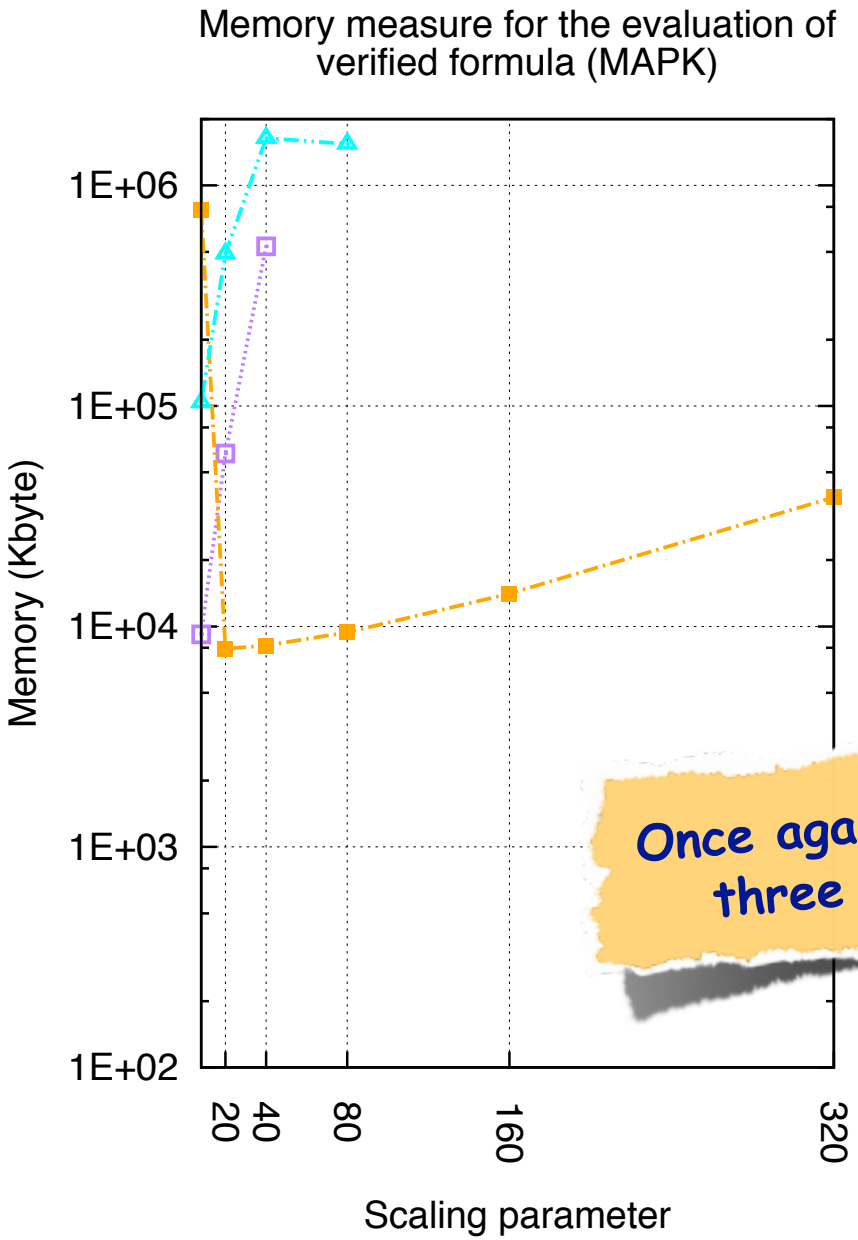


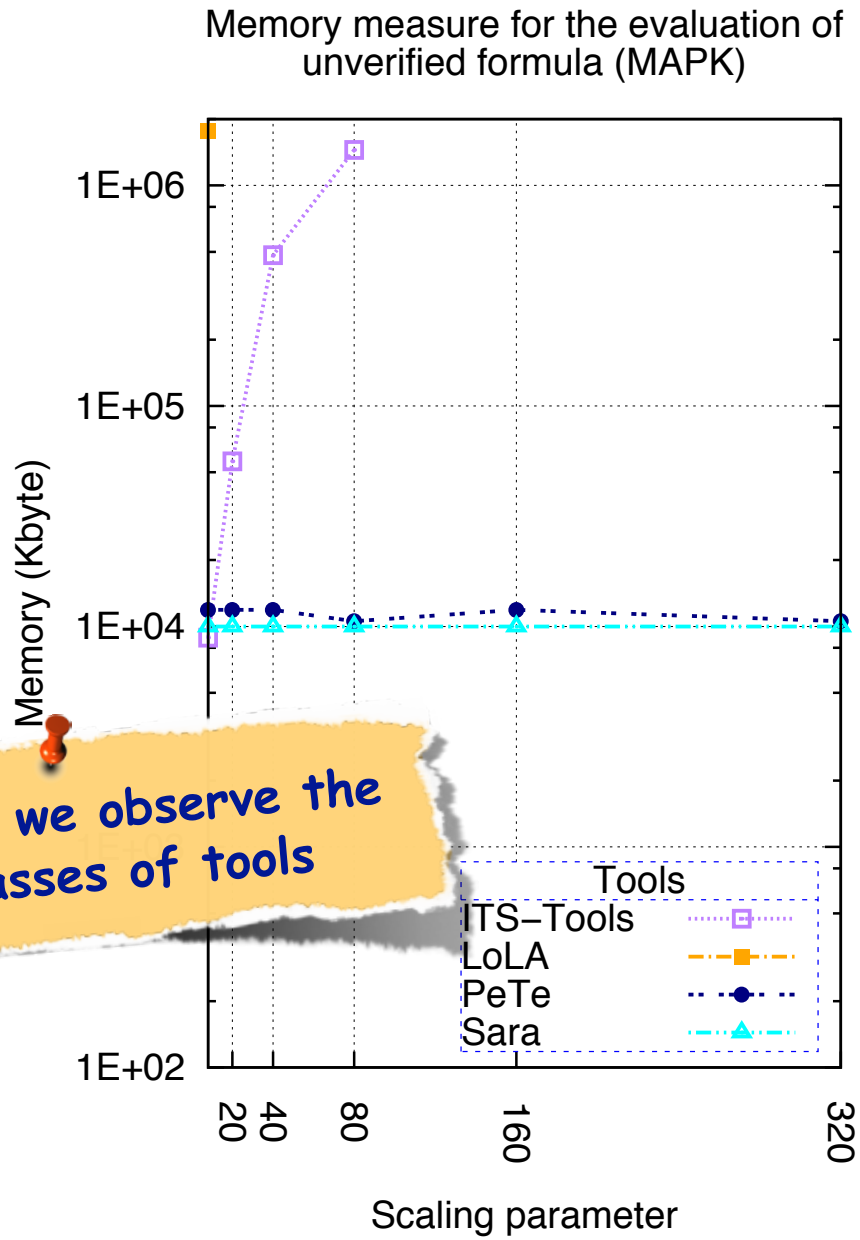
MAPK, Reachability Analysis





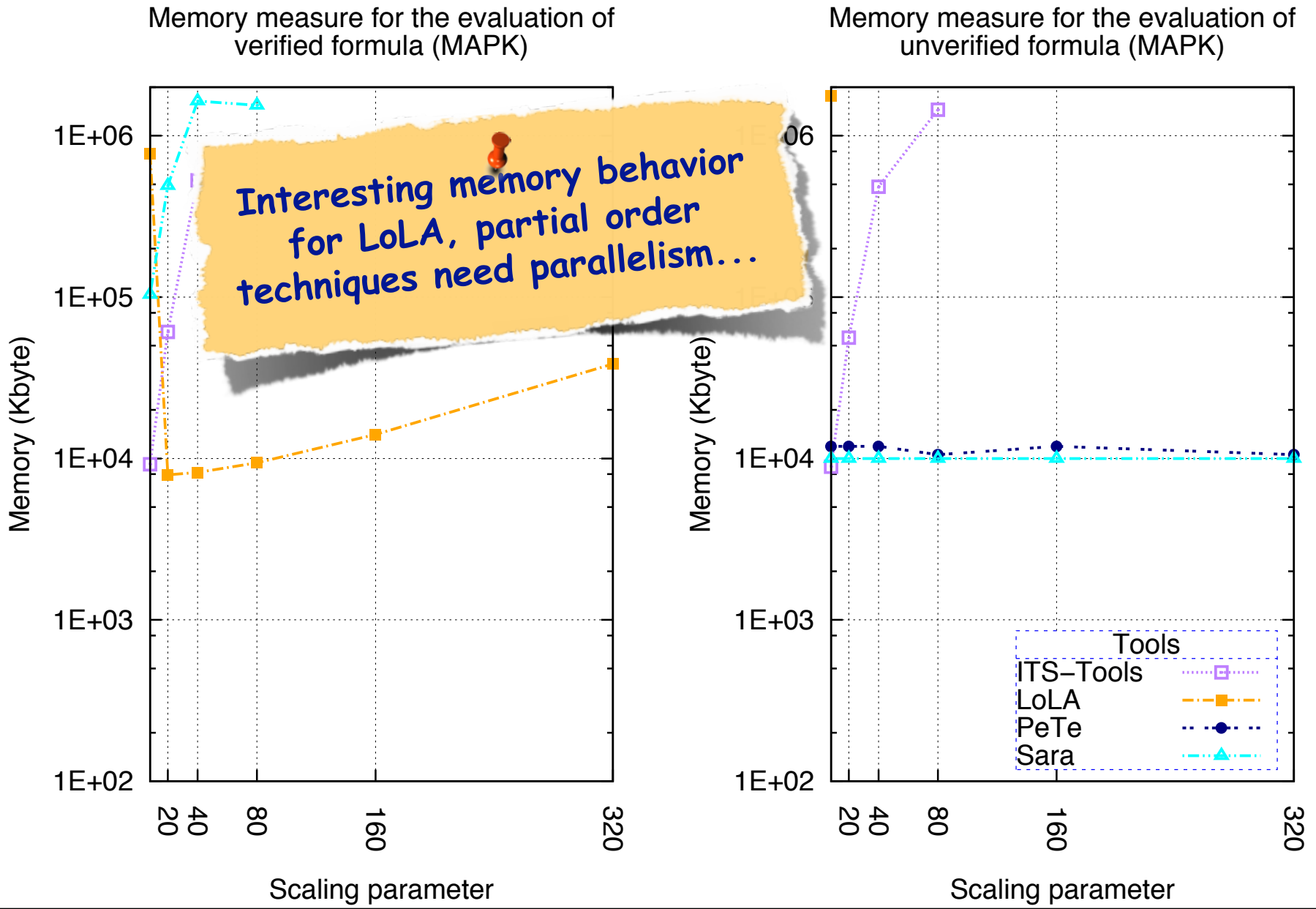
MAPK, Reachability Analysis





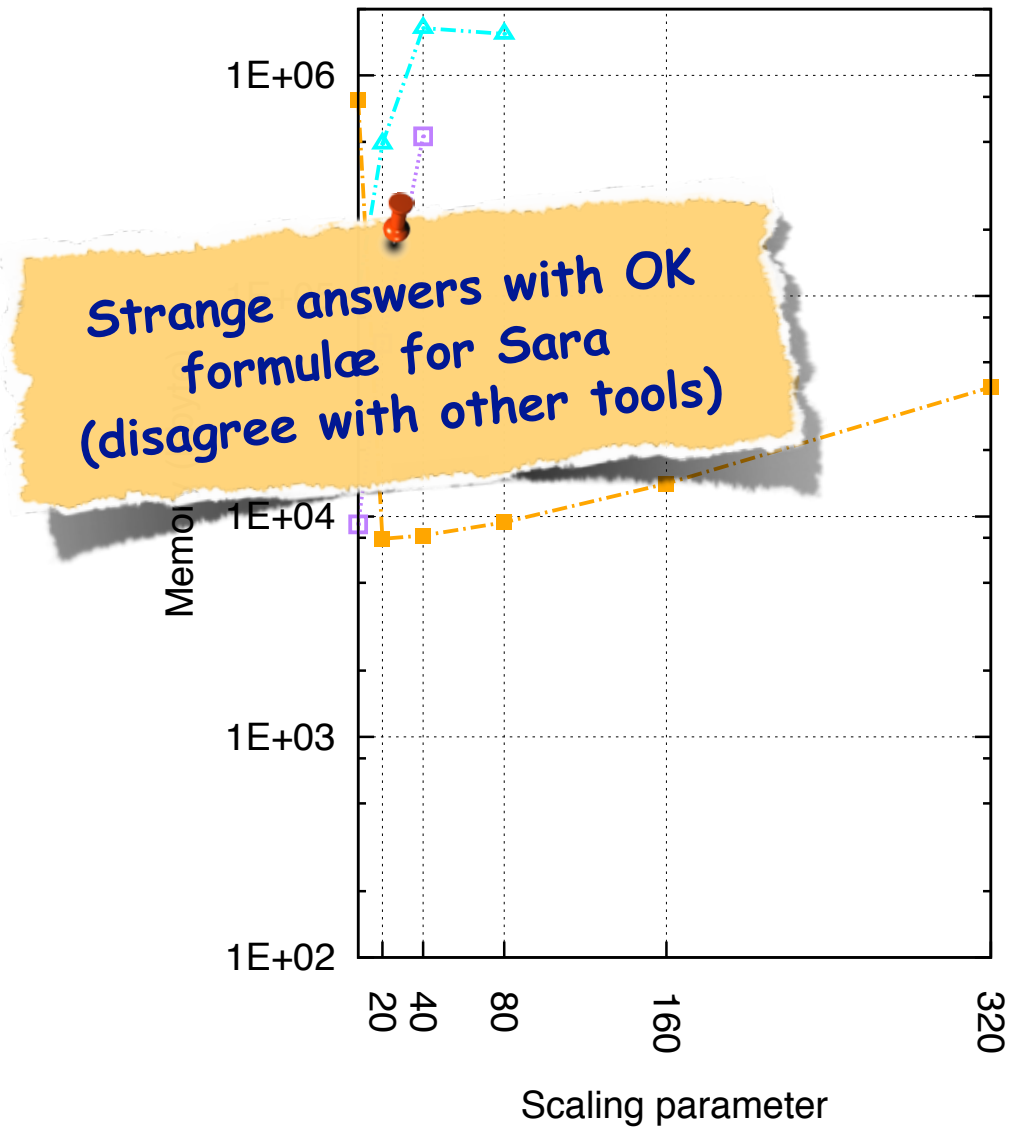
Once again, we observe the three classes of tools

MAPK, Reachability Analysis



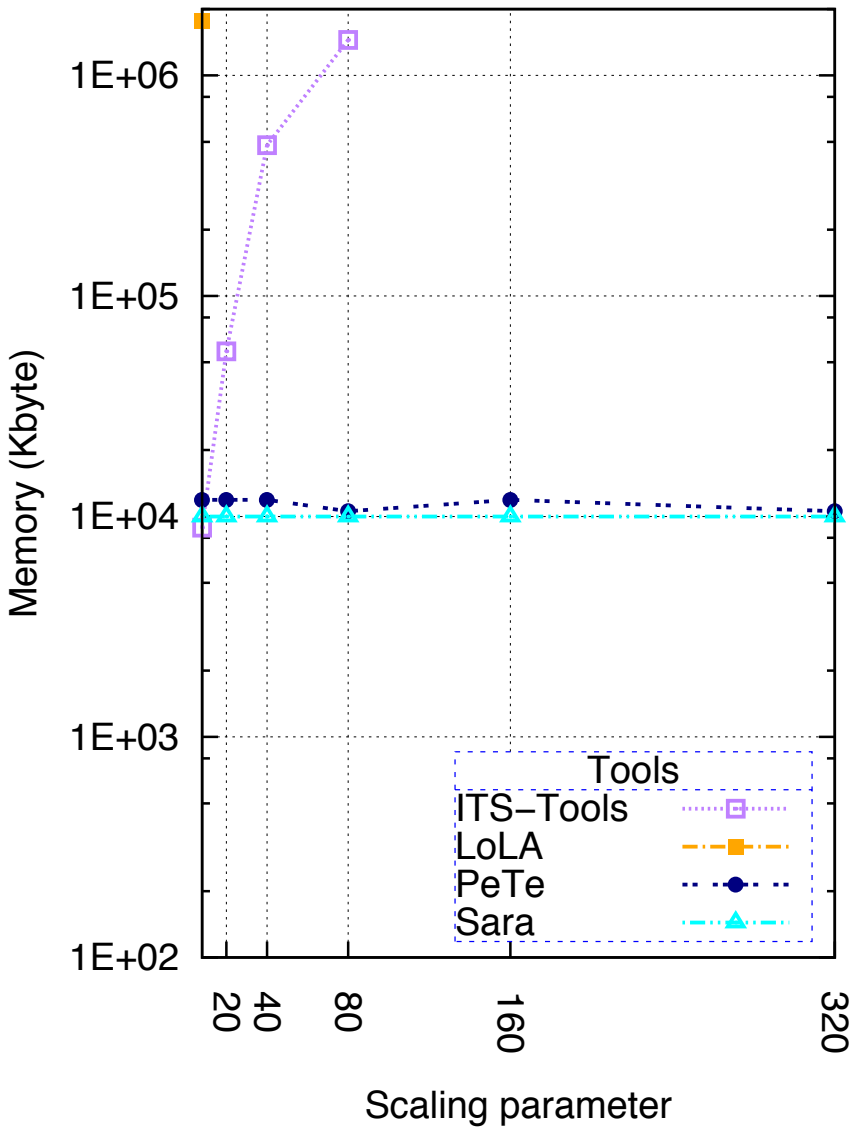
MAPK, Reachability Analysis

Memory measure for the evaluation of verified formula (MAPK)

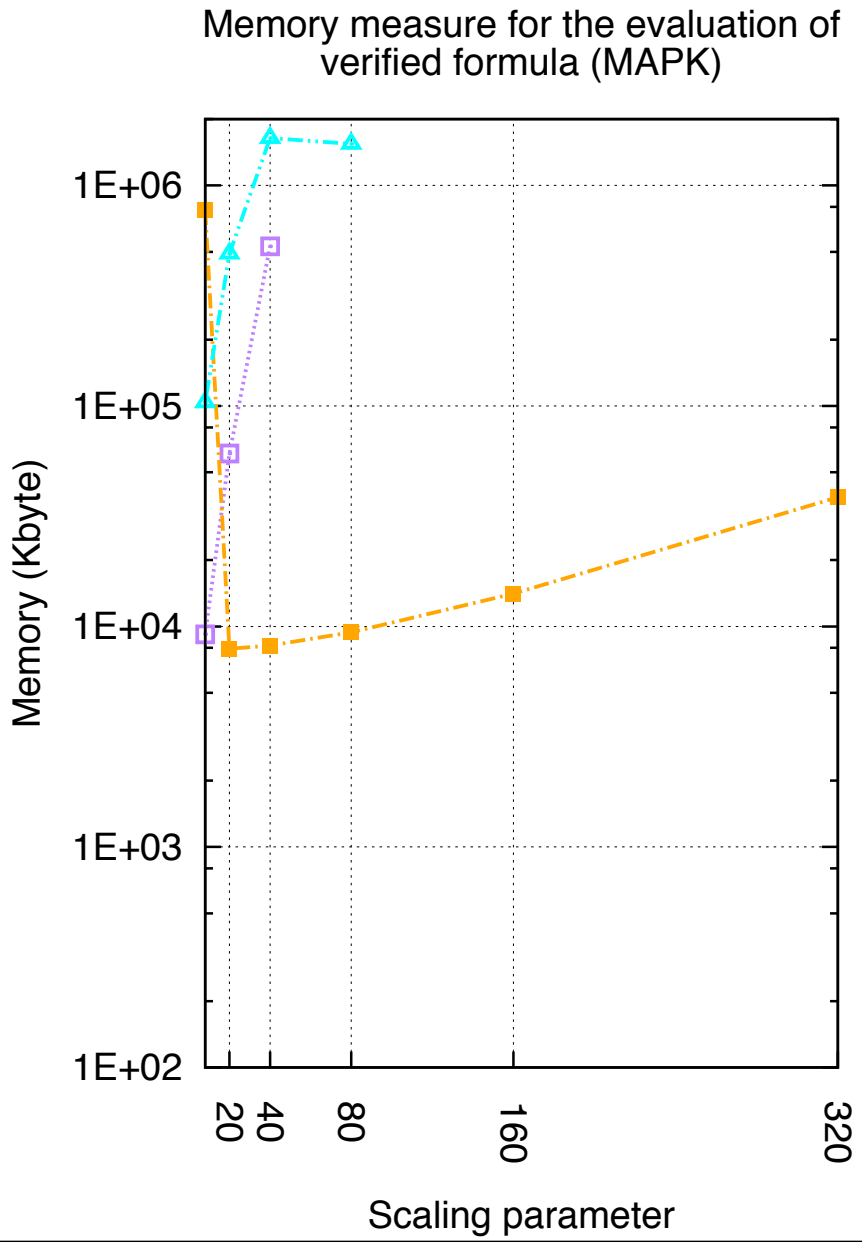


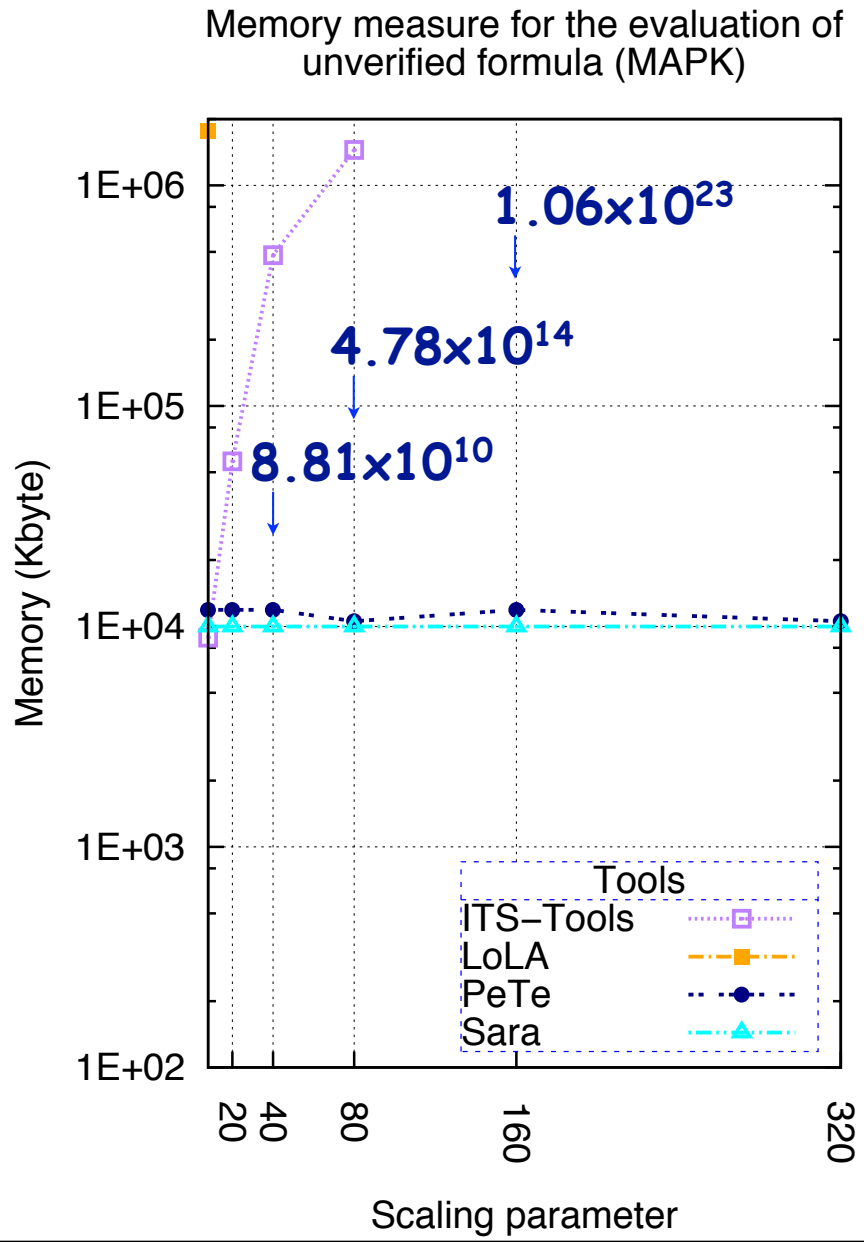
Strange answers with OK formulæ for Sara (disagree with other tools)

Memory measure for the evaluation of unverified formula (MAPK)

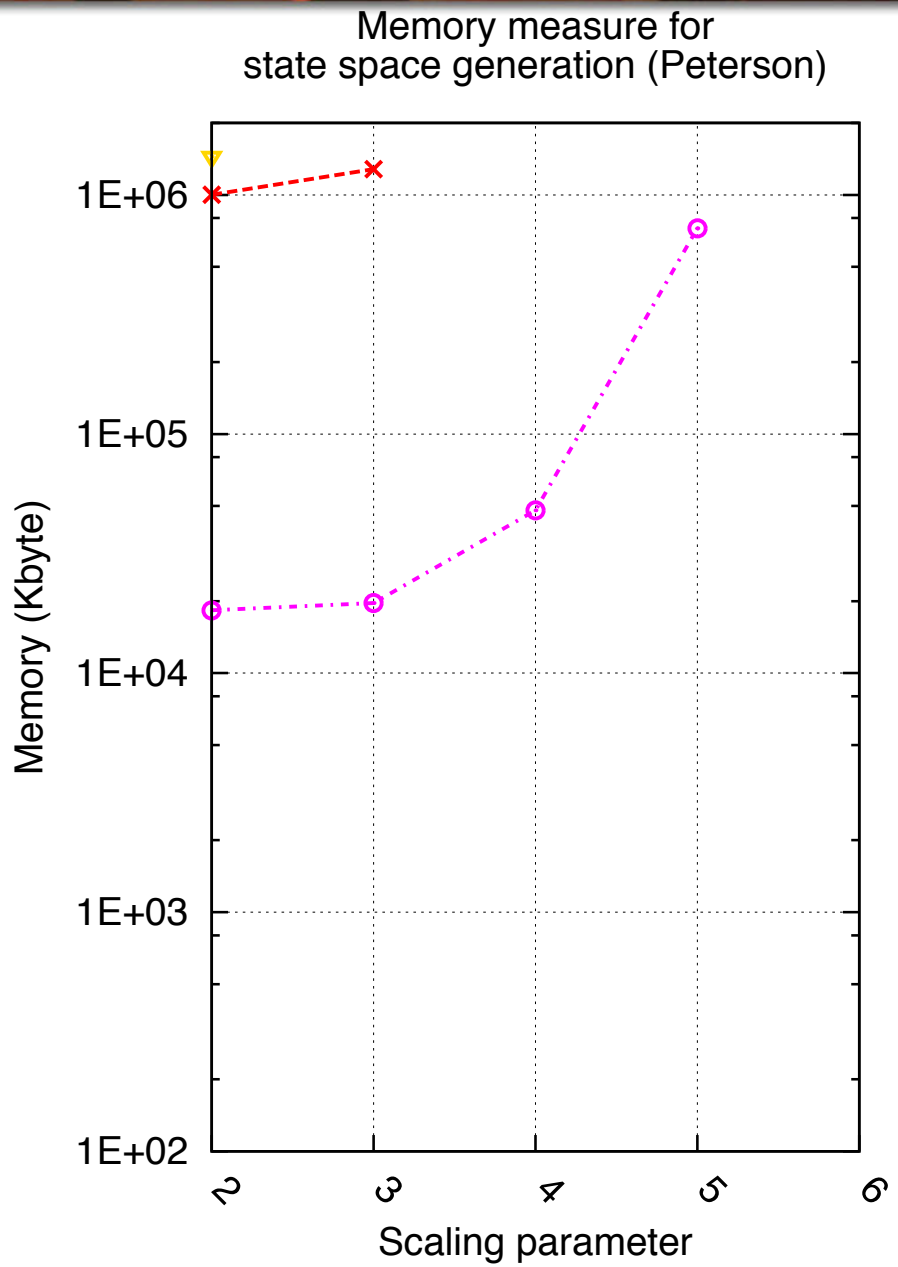


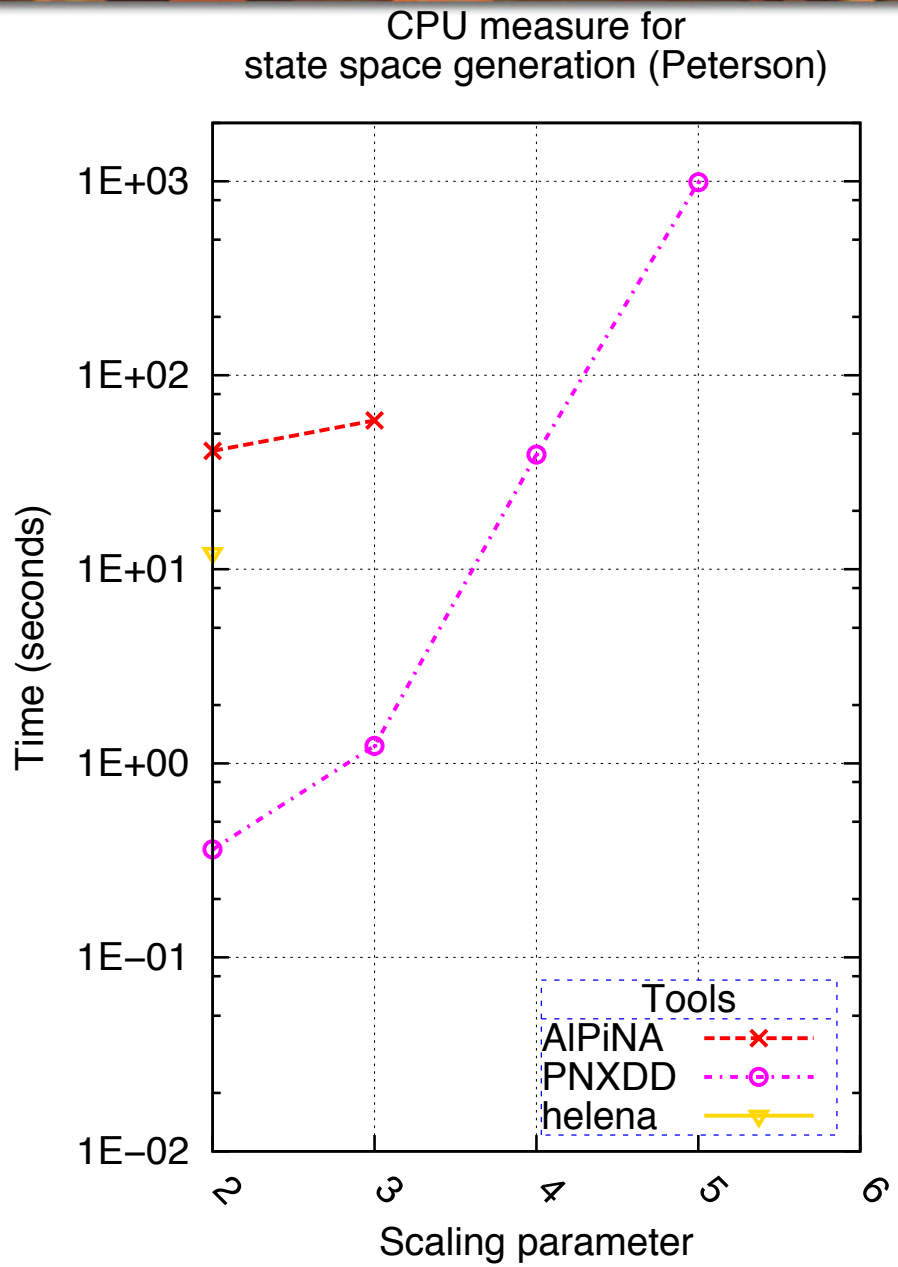
MAPK, Reachability Analysis





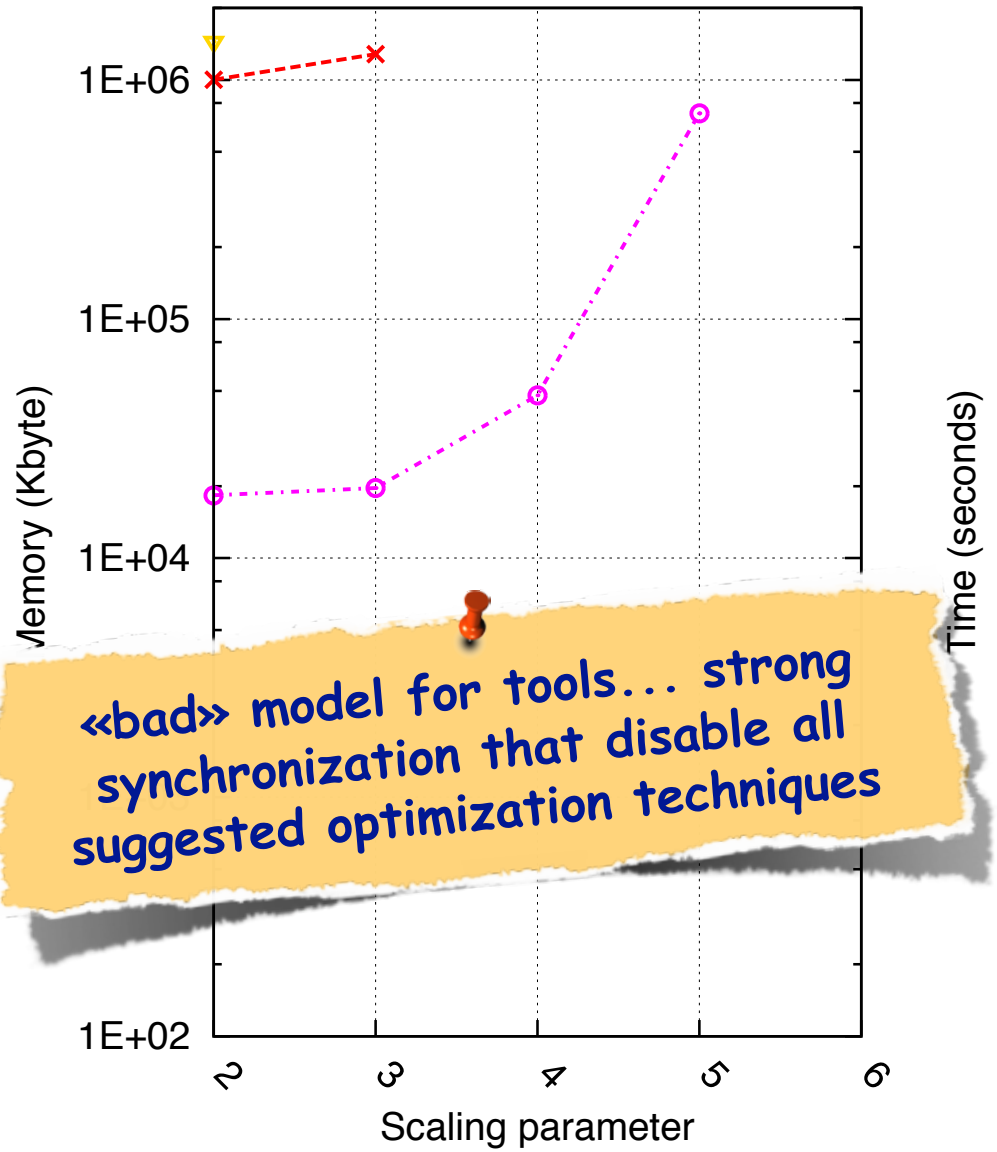
Peterson, State space generation





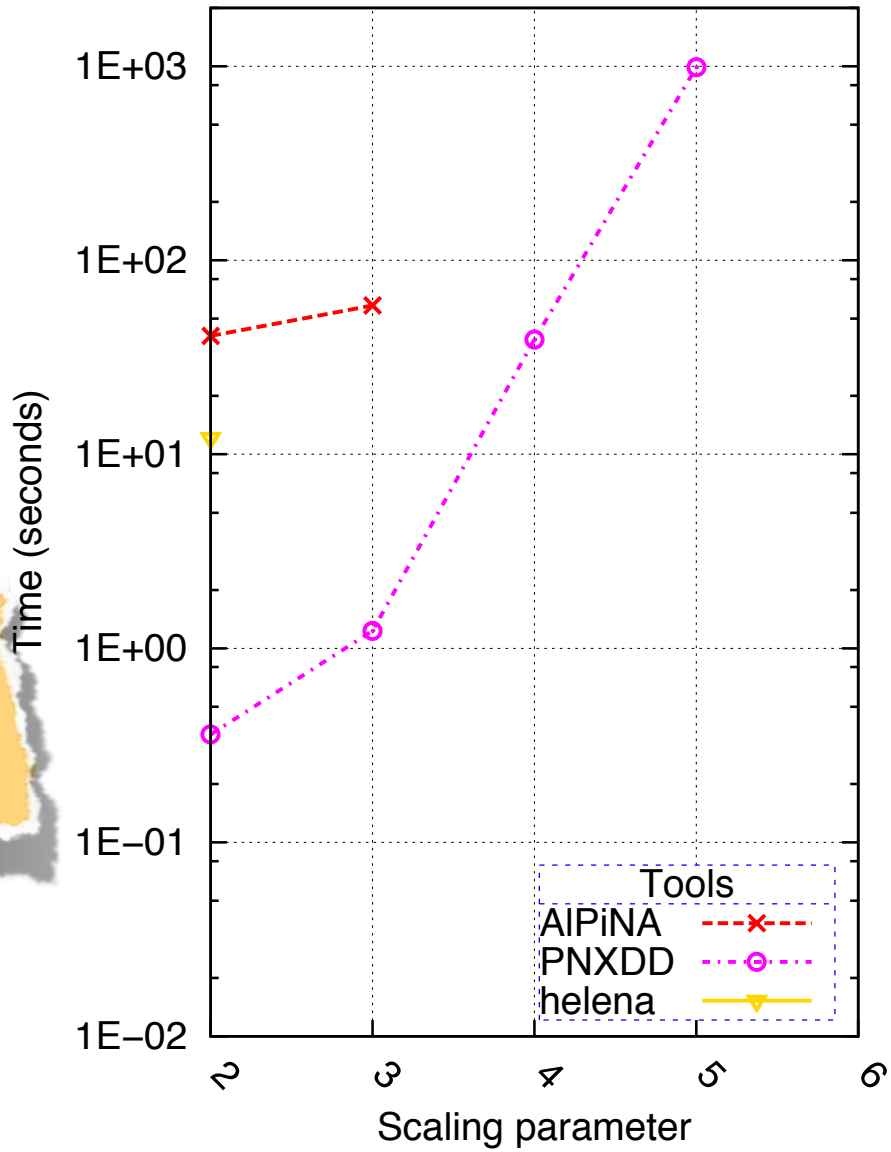
Peterson, State space generation

Memory measure for
state space generation (Peterson)



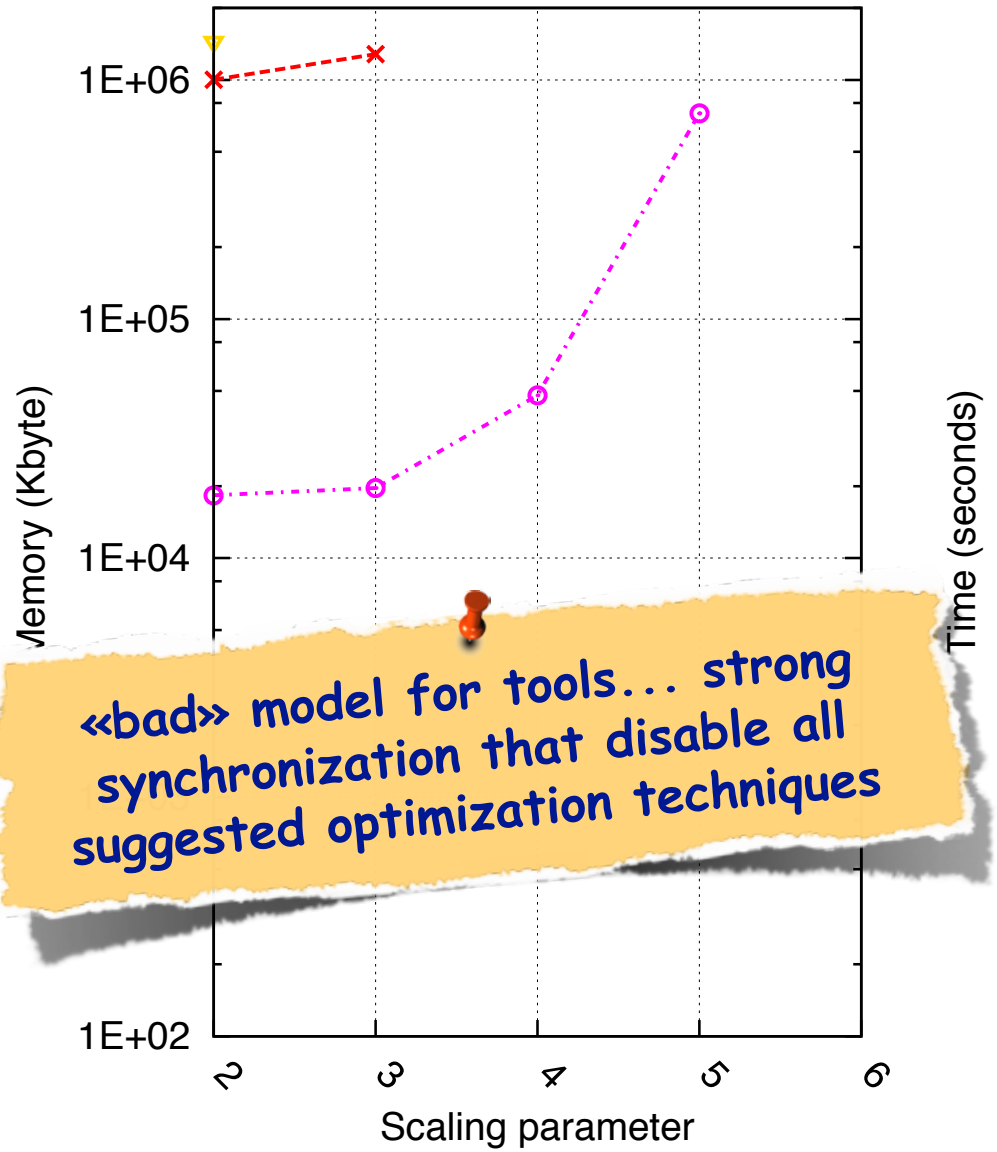
«bad» model for tools... strong synchronization that disable all suggested optimization techniques

CPU measure for
state space generation (Peterson)



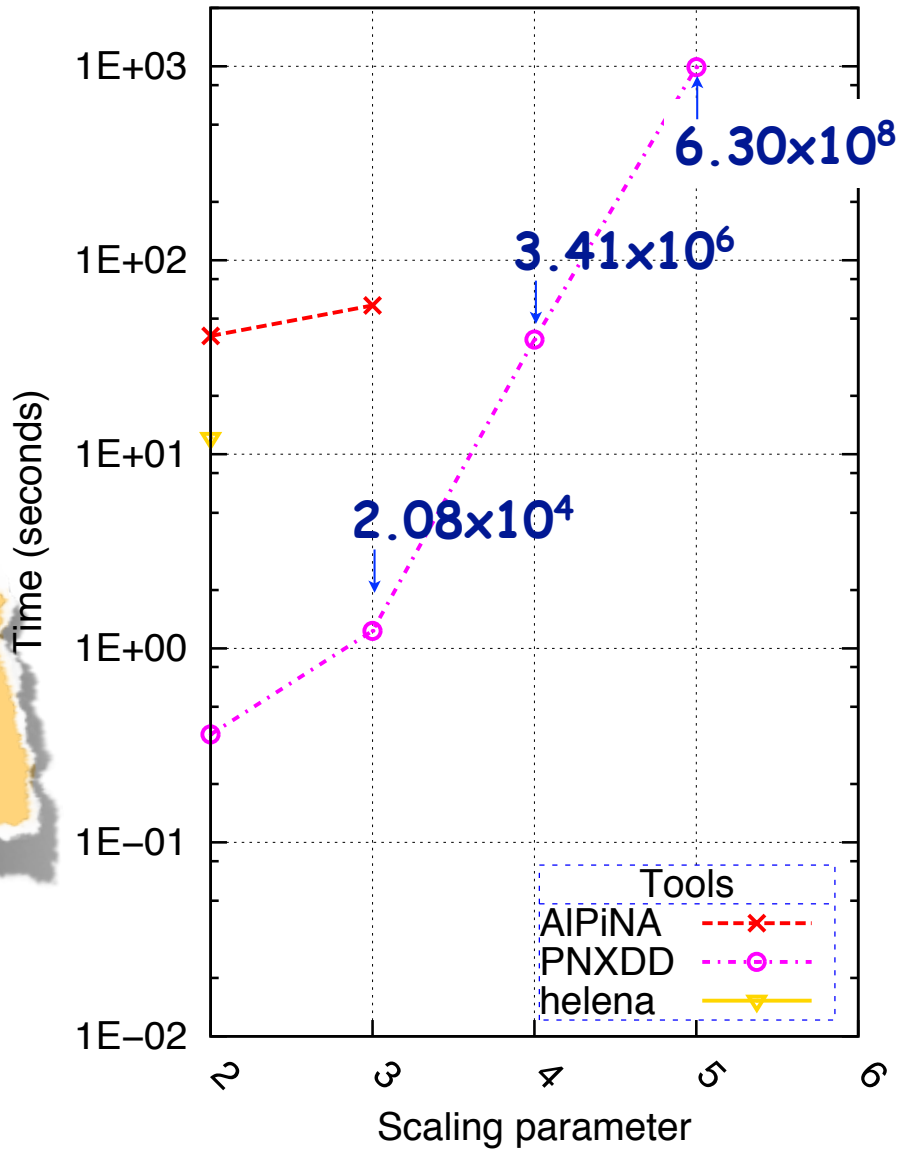
Peterson, State space generation

Memory measure for
state space generation (Peterson)

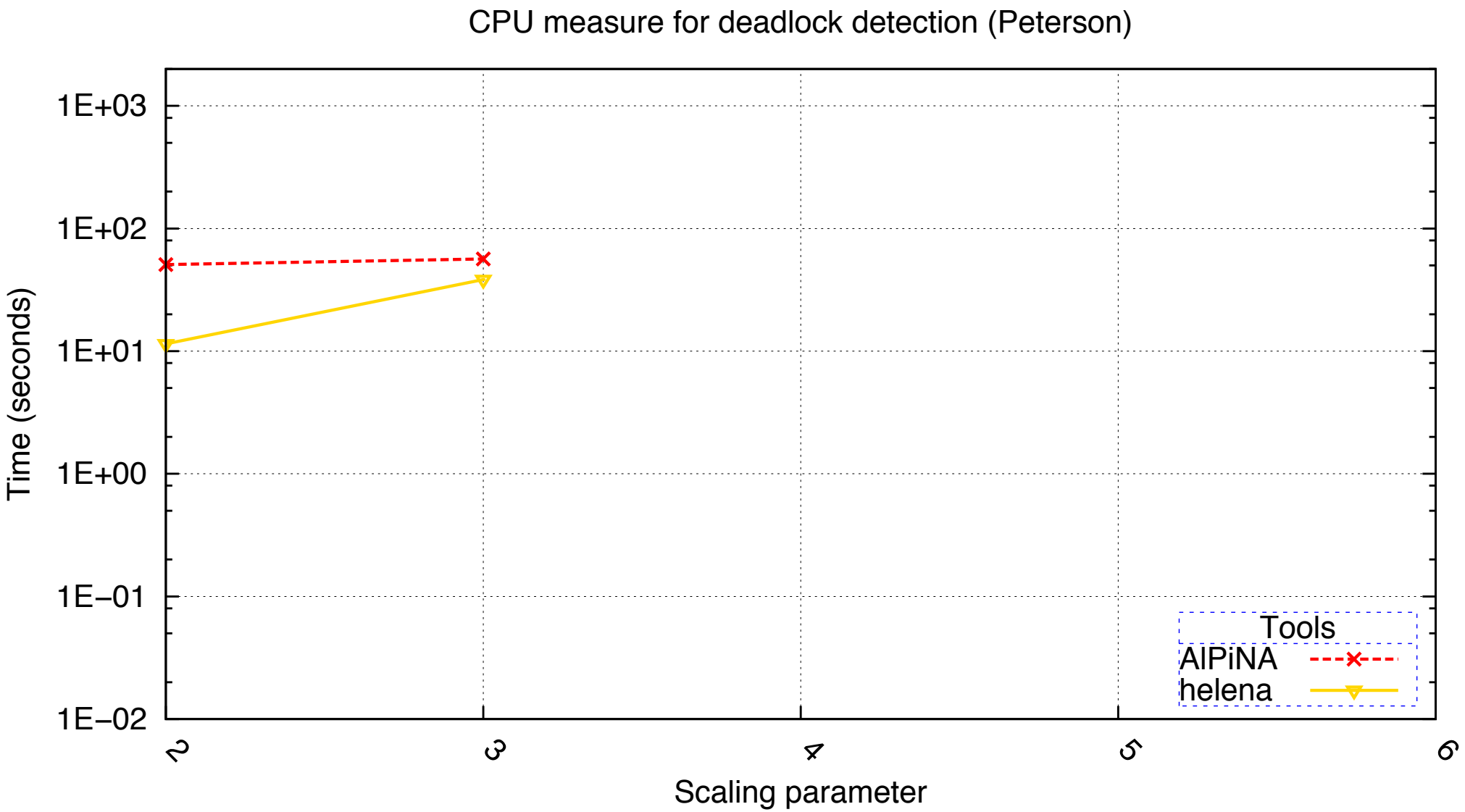


«bad» model for tools... strong synchronization that disable all suggested optimization techniques

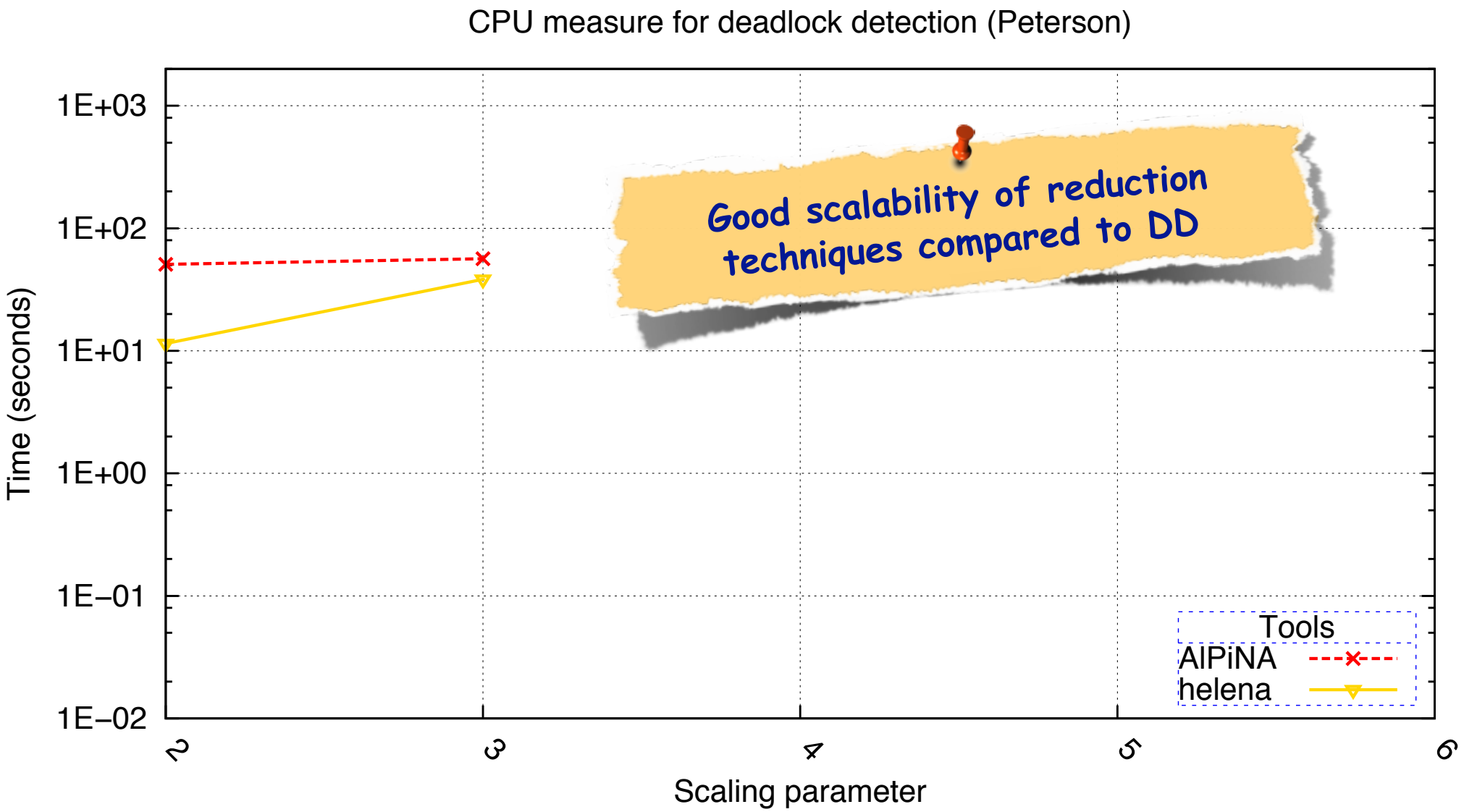
CPU measure for
state space generation (Peterson)



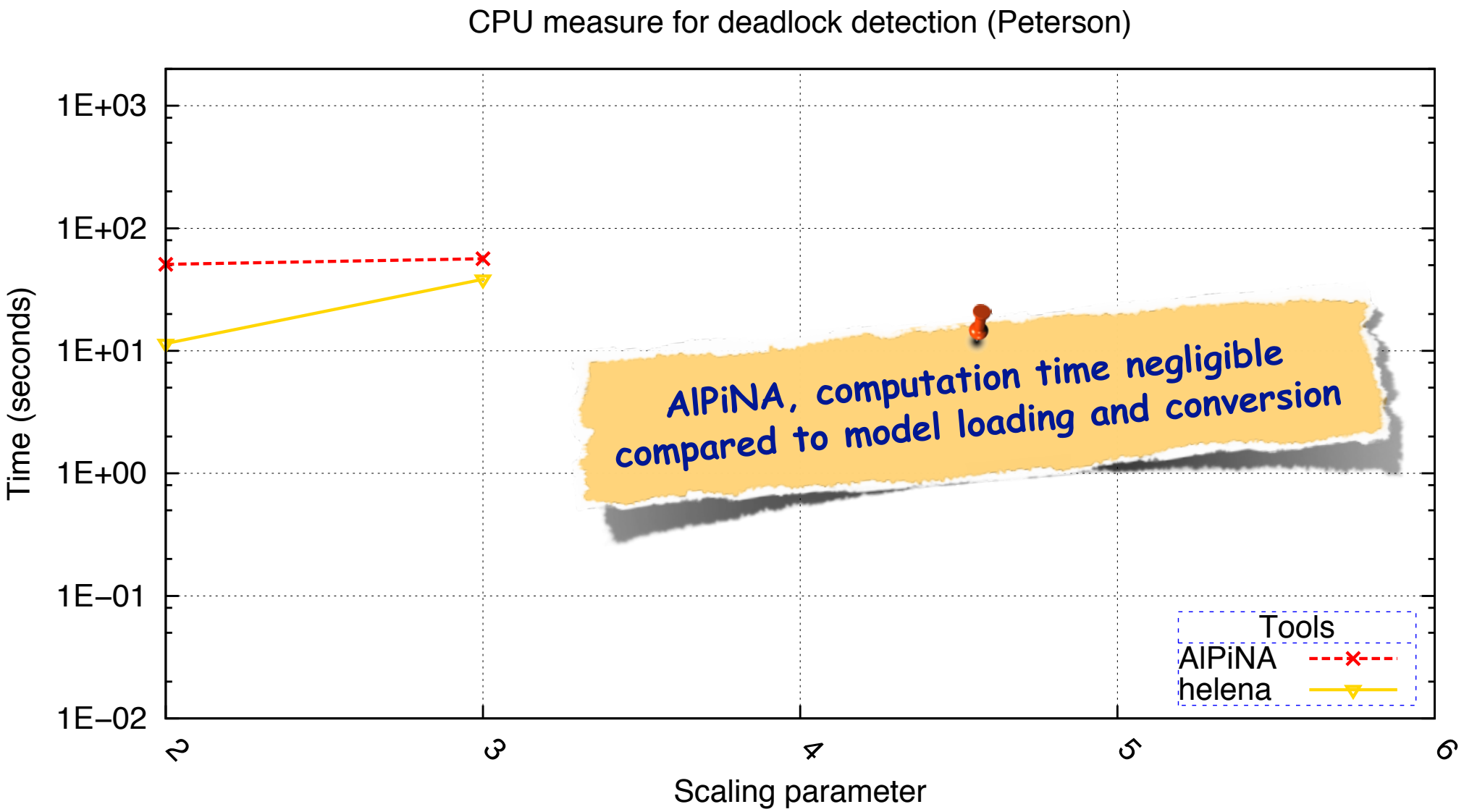
Peterson, Deadlock Detection



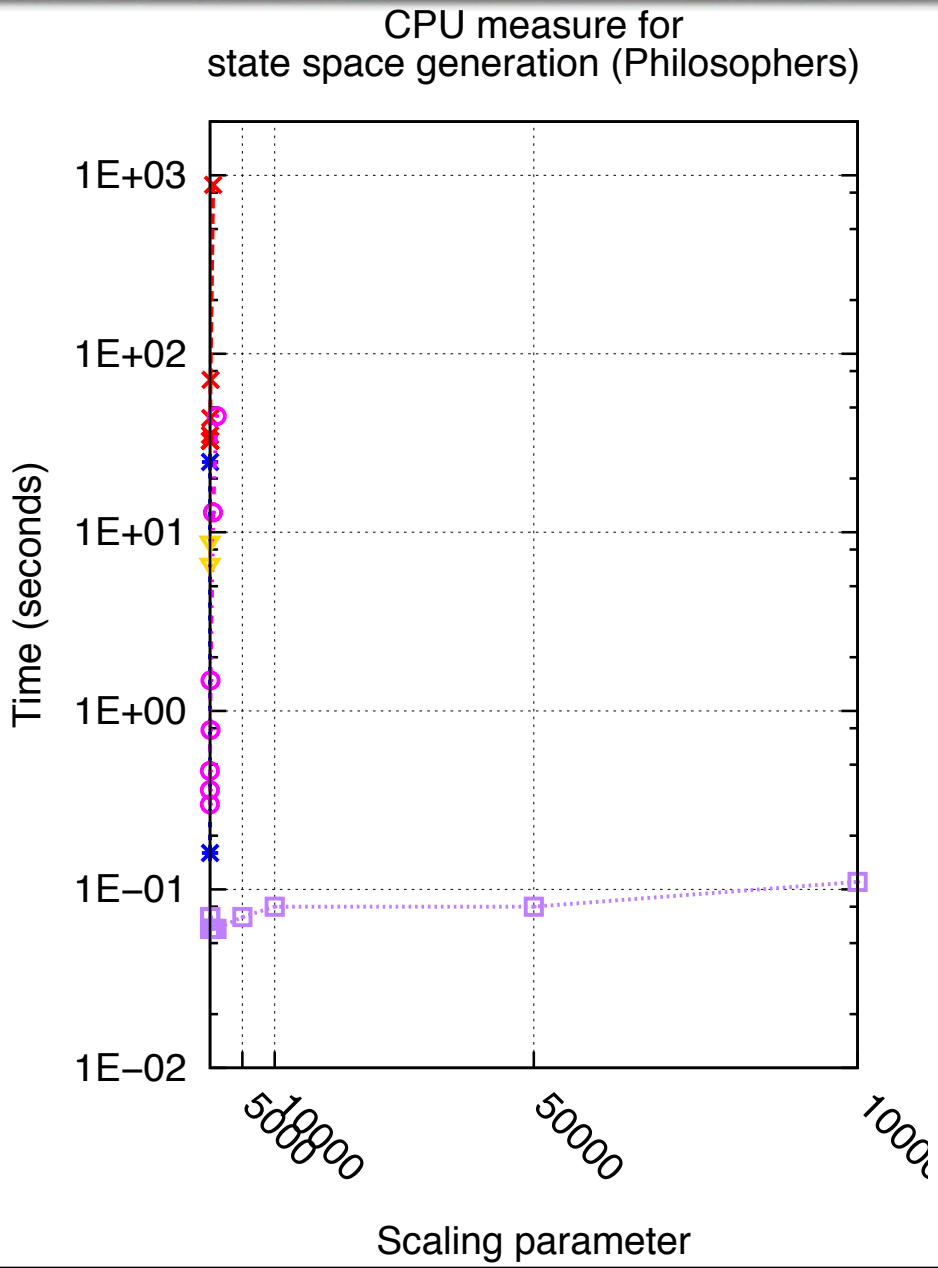
Peterson, Deadlock Detection

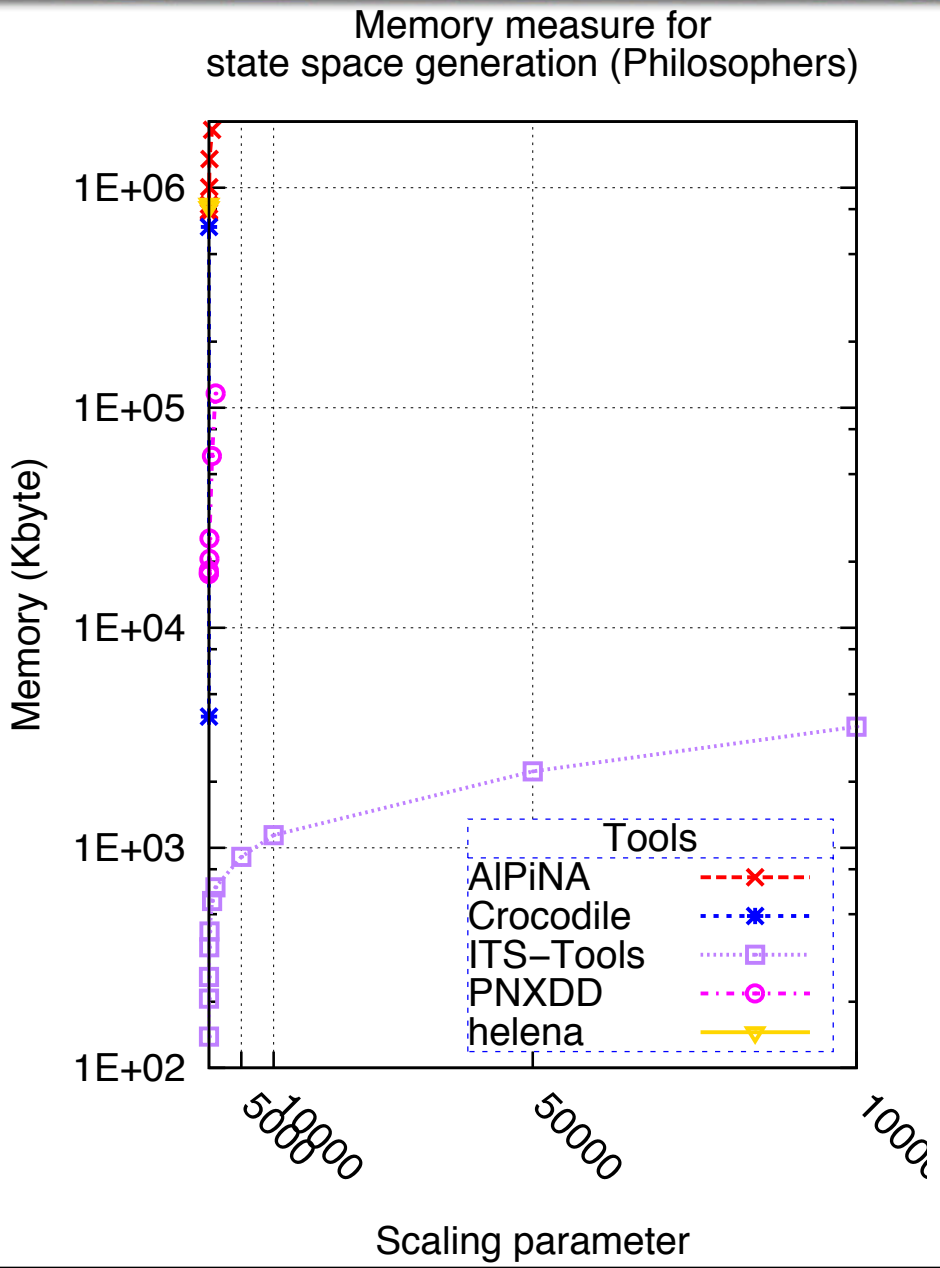


Peterson, Deadlock Detection

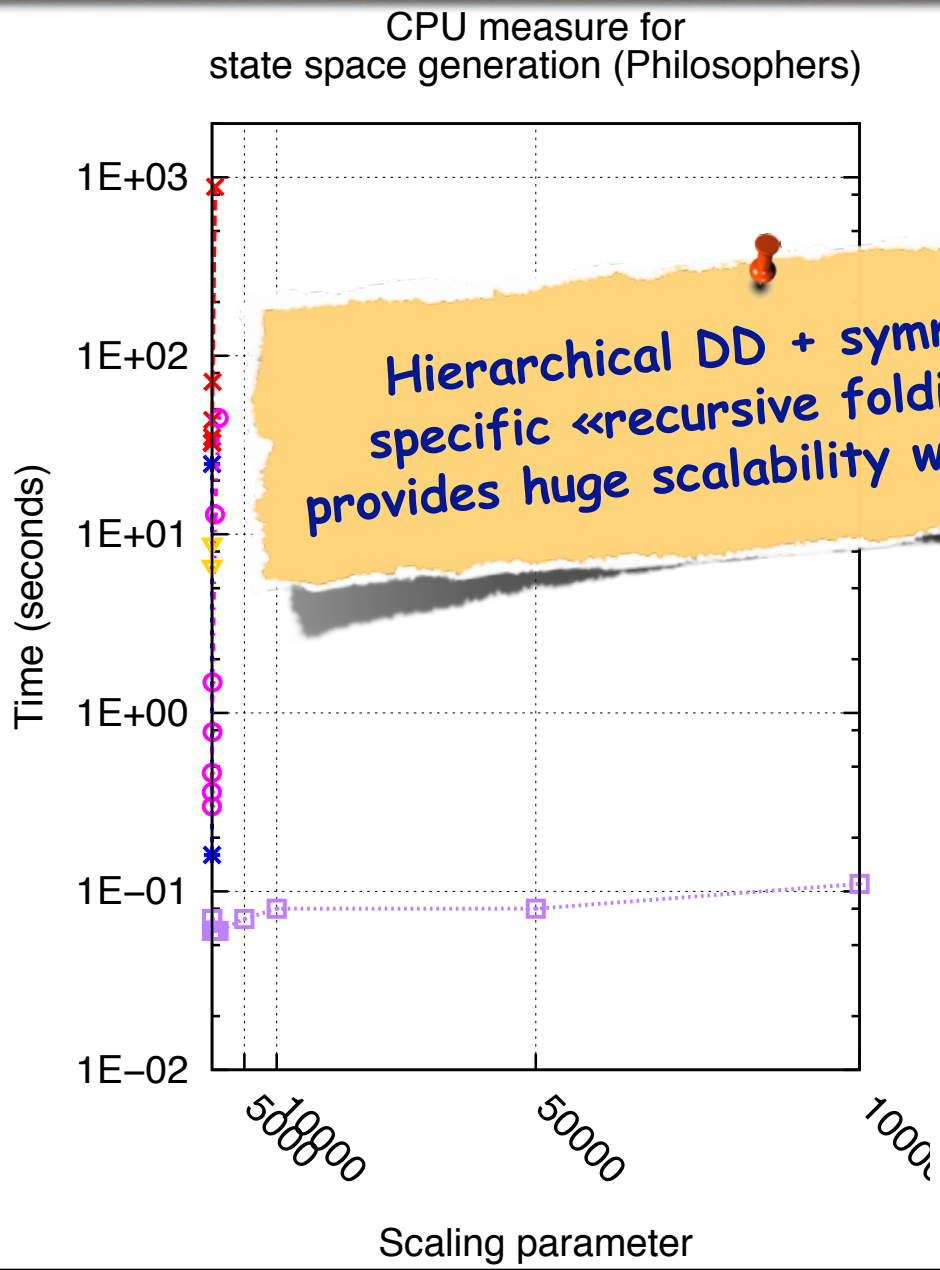


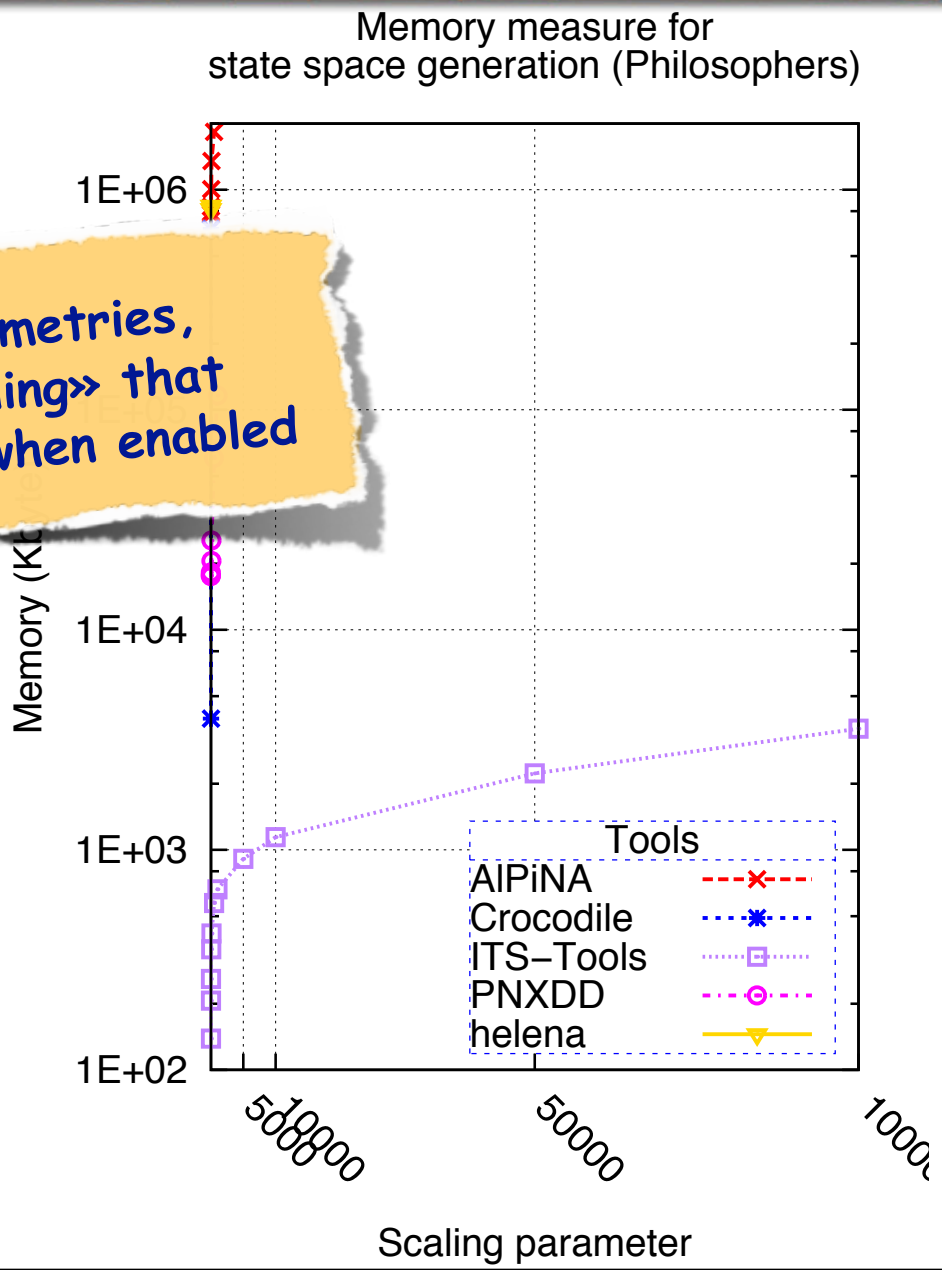
Philosopher, State Space Generation





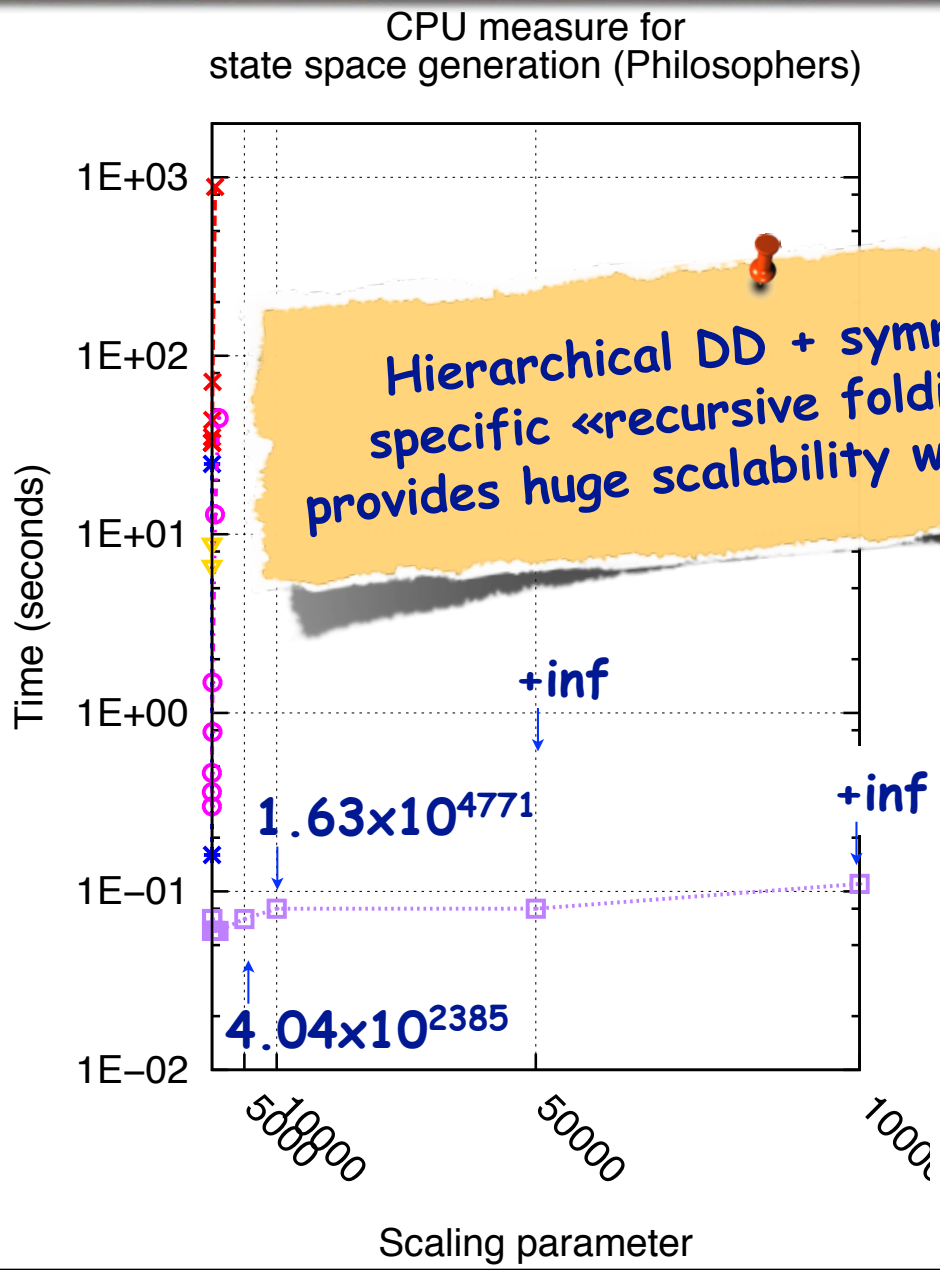
Philosopher, State Space Generation

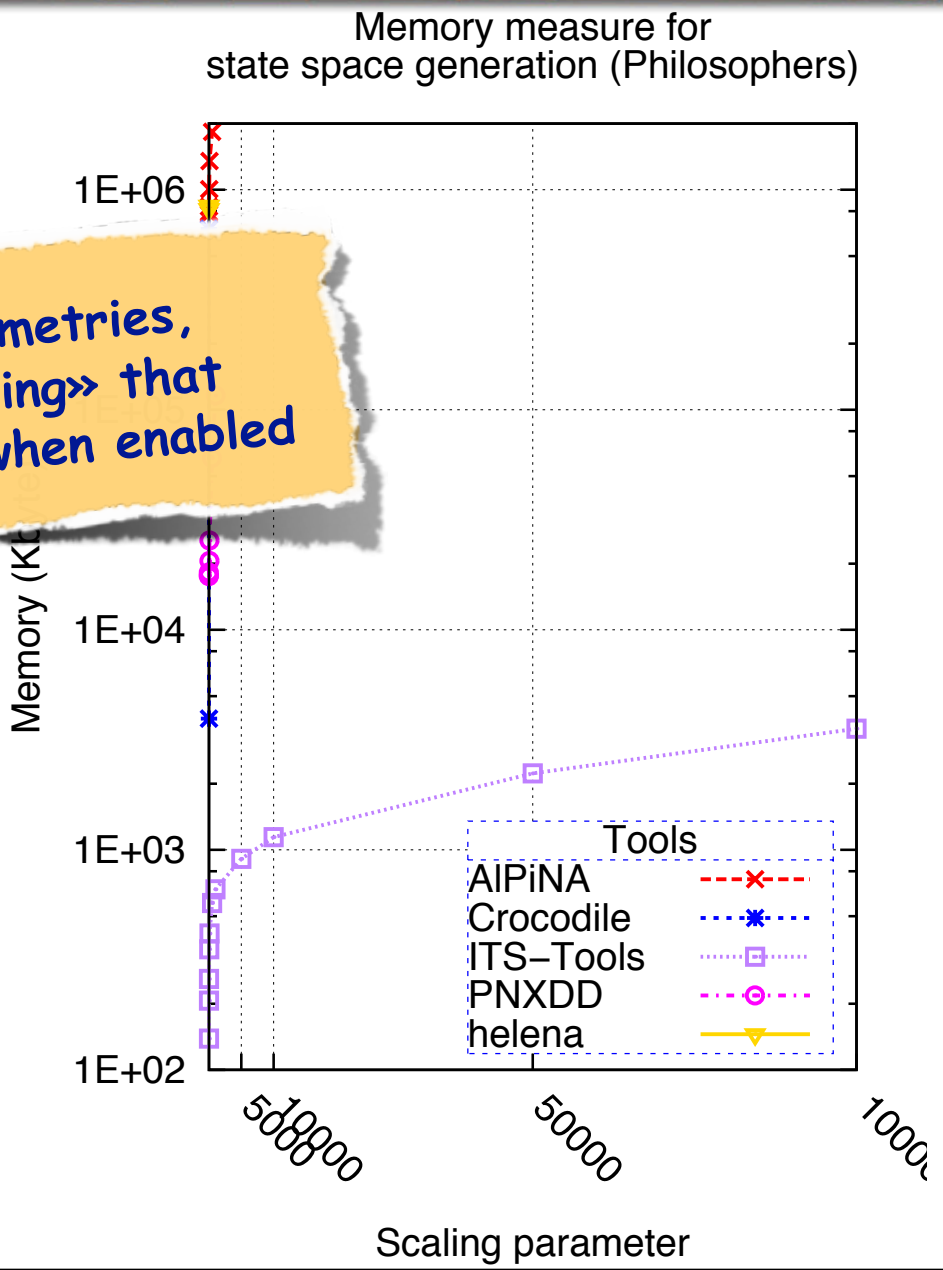




Hierarchical DD + symmetries, specific «recursive folding» that provides huge scalability when enabled

Philosopher, State Space Generation

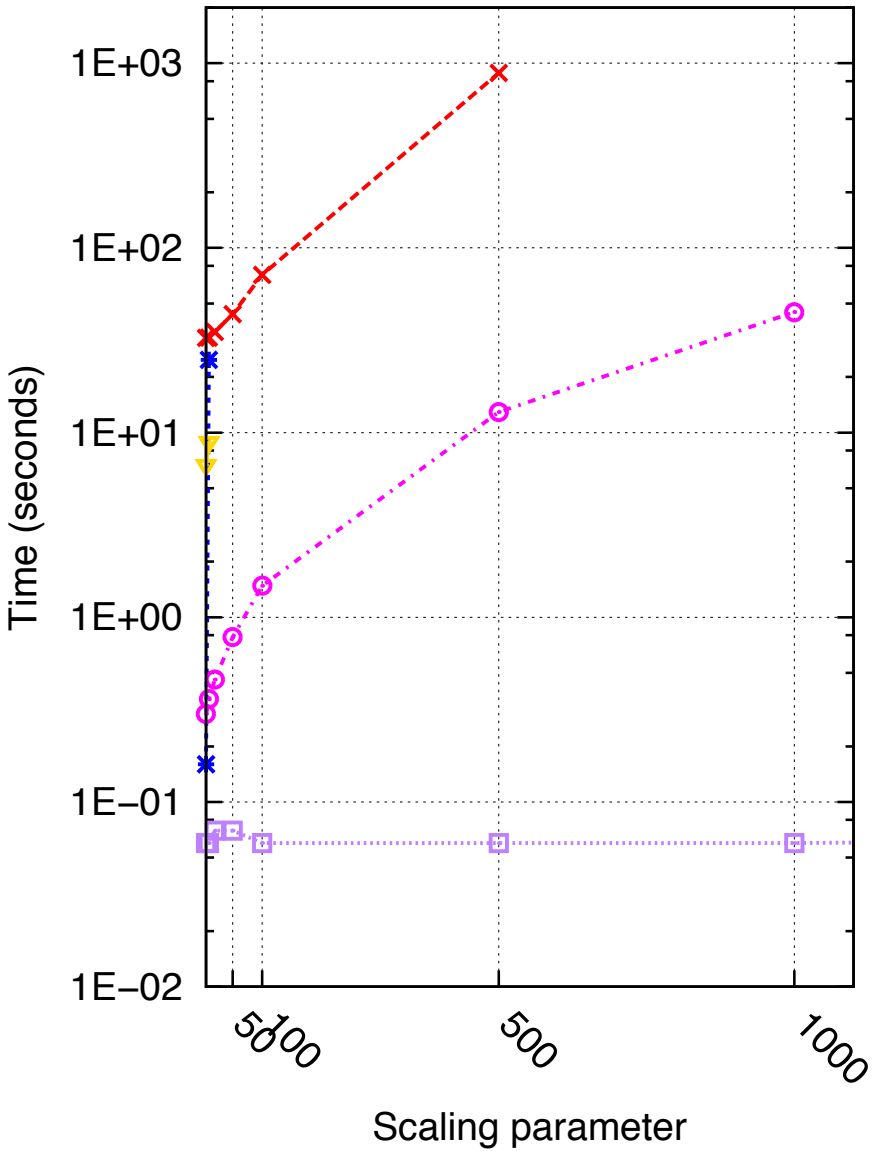




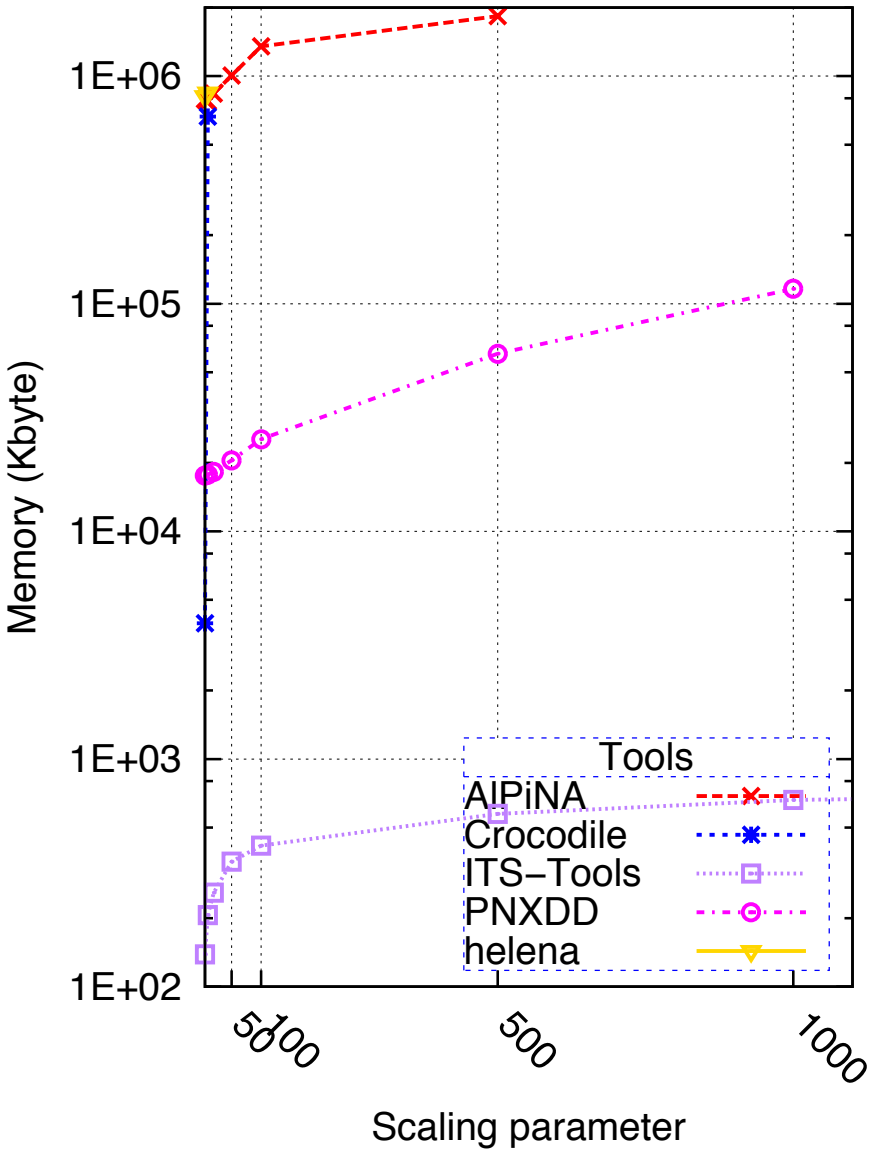
Hierarchical DD + symmetries, specific «recursive folding» that provides huge scalability when enabled

Philosopher, State Space Generation

CPU measure for
state space generation (Philosophers)

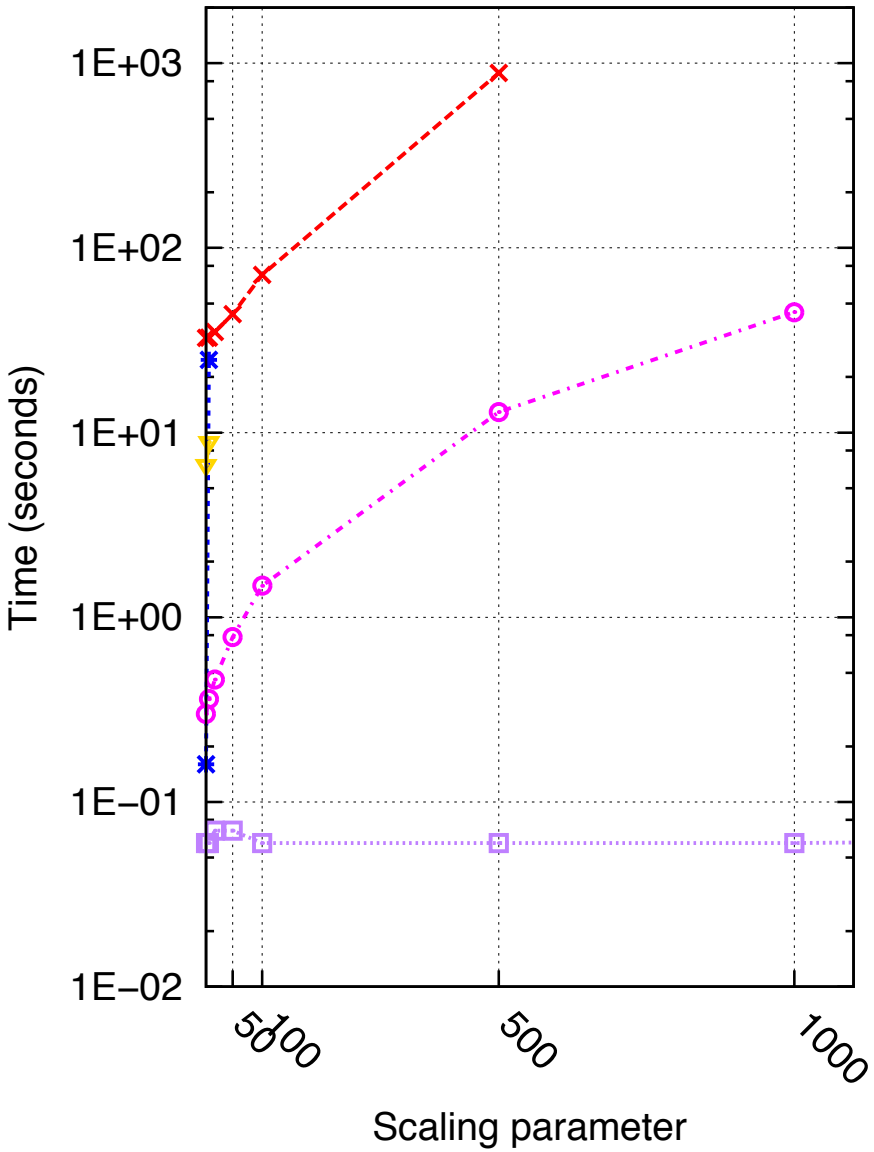


Memory measure for
state space generation (Philosophers)

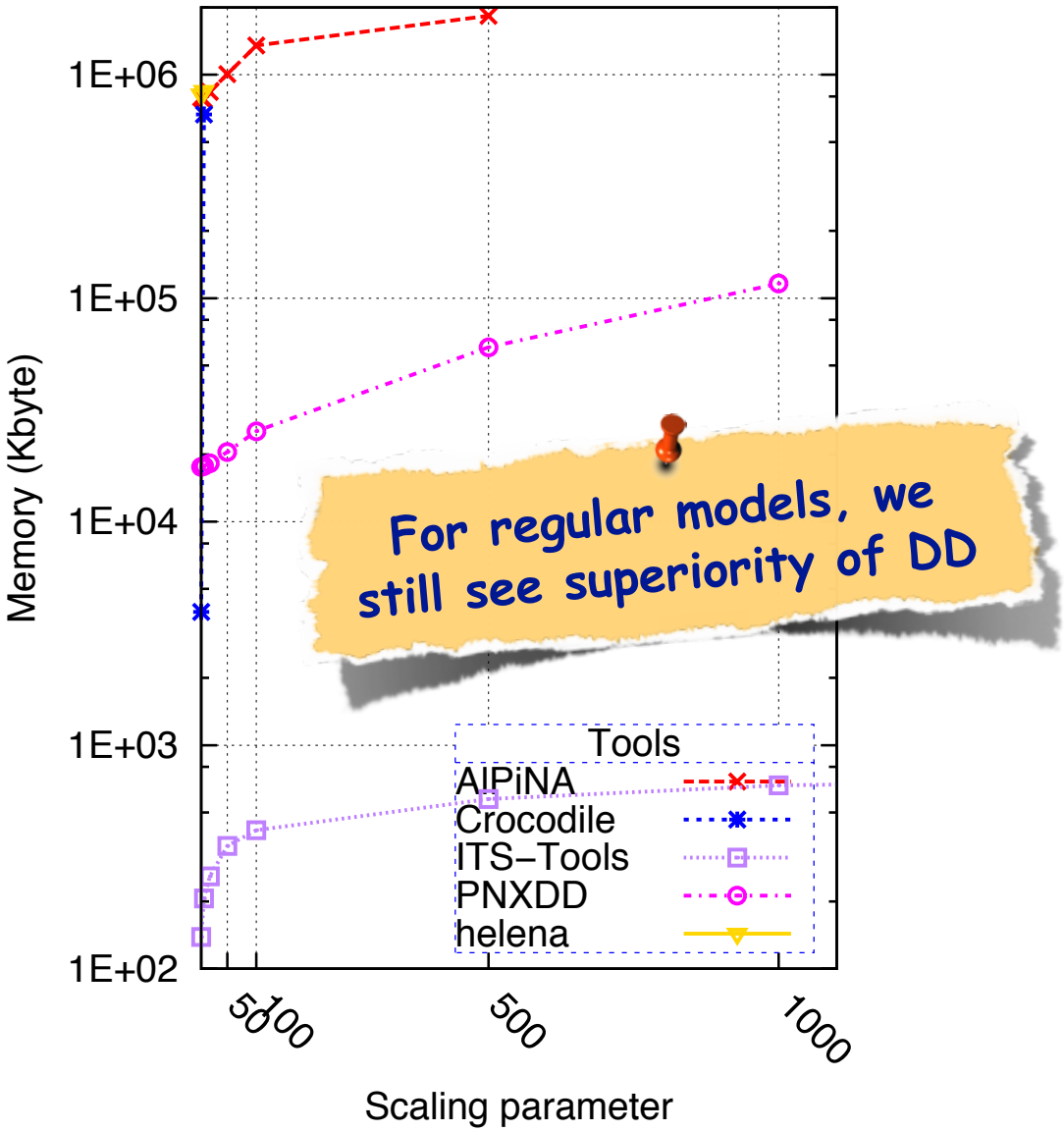


Philosopher, State Space Generation

CPU measure for
state space generation (Philosophers)



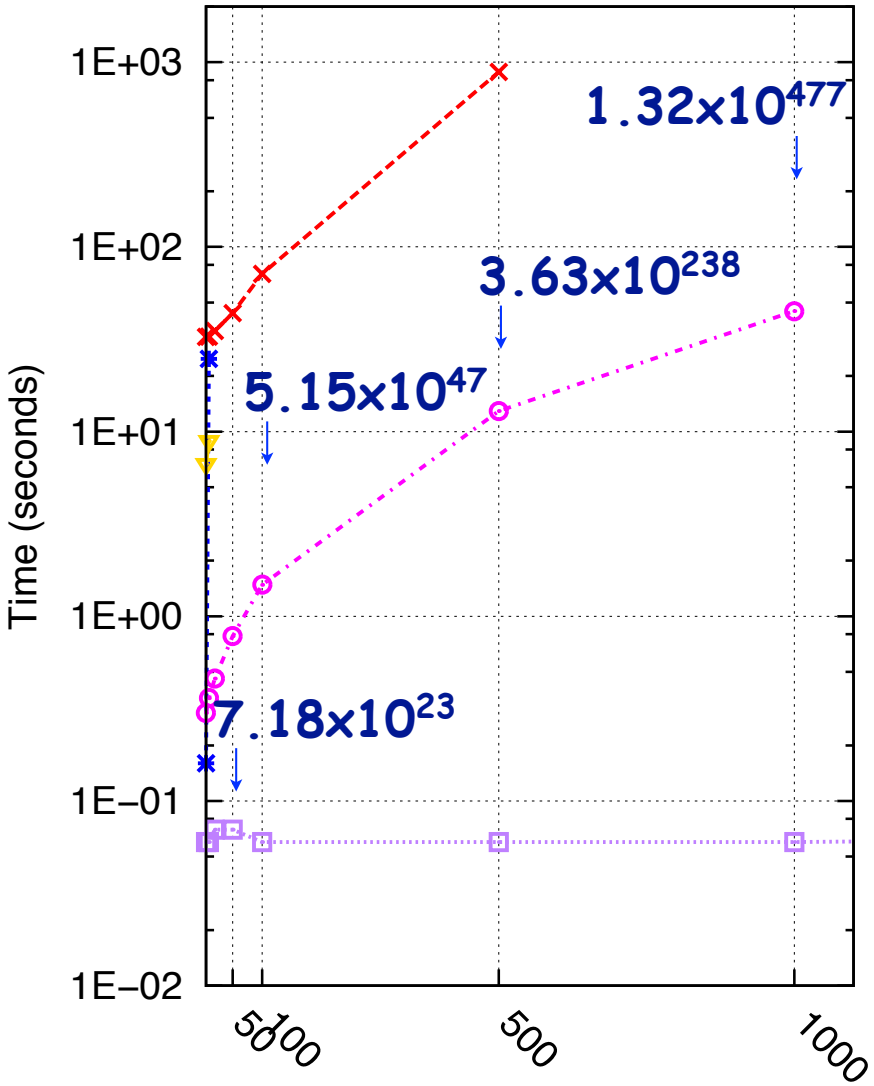
Memory measure for
state space generation (Philosophers)



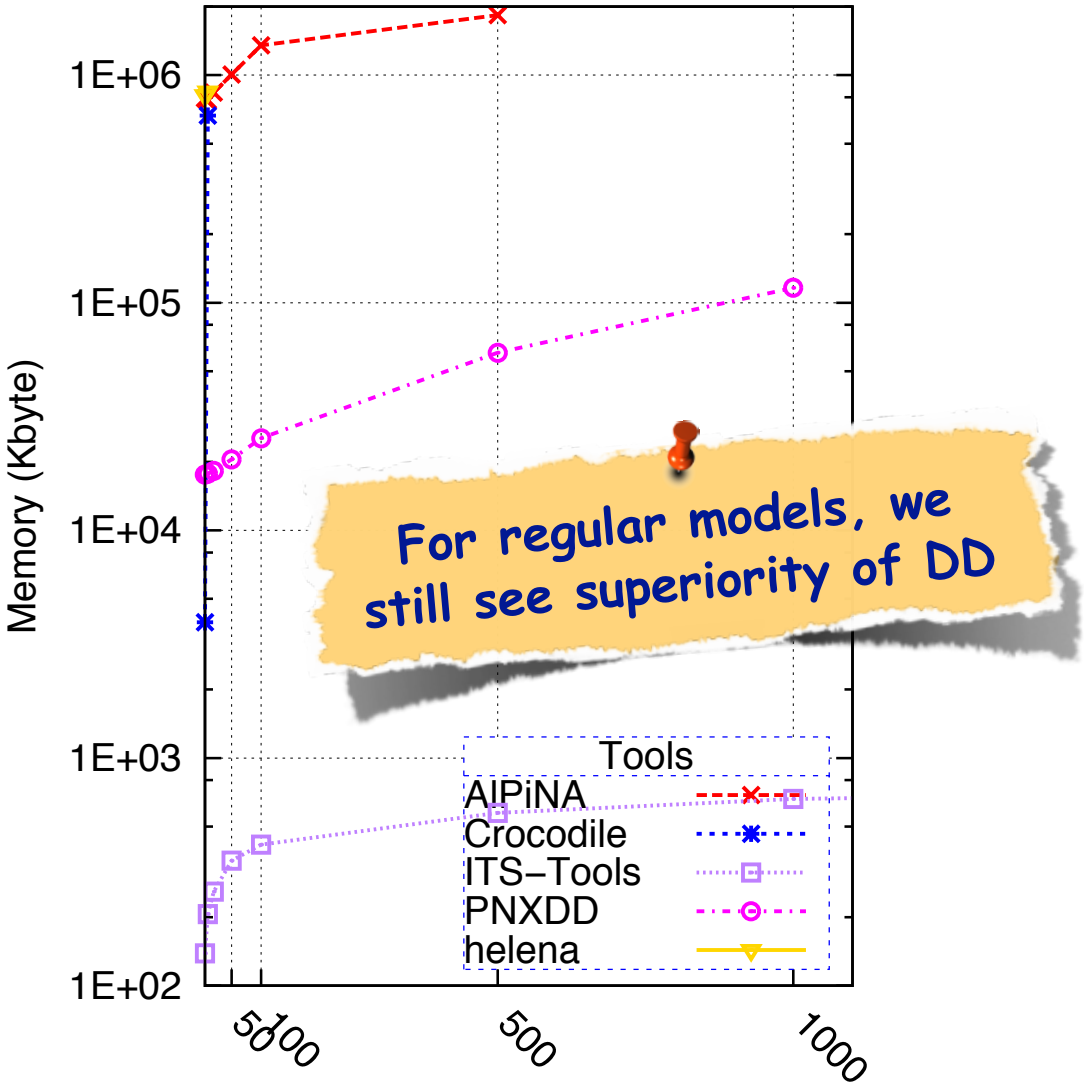
For regular models, we still see superiority of DD

Philosopher, State Space Generation

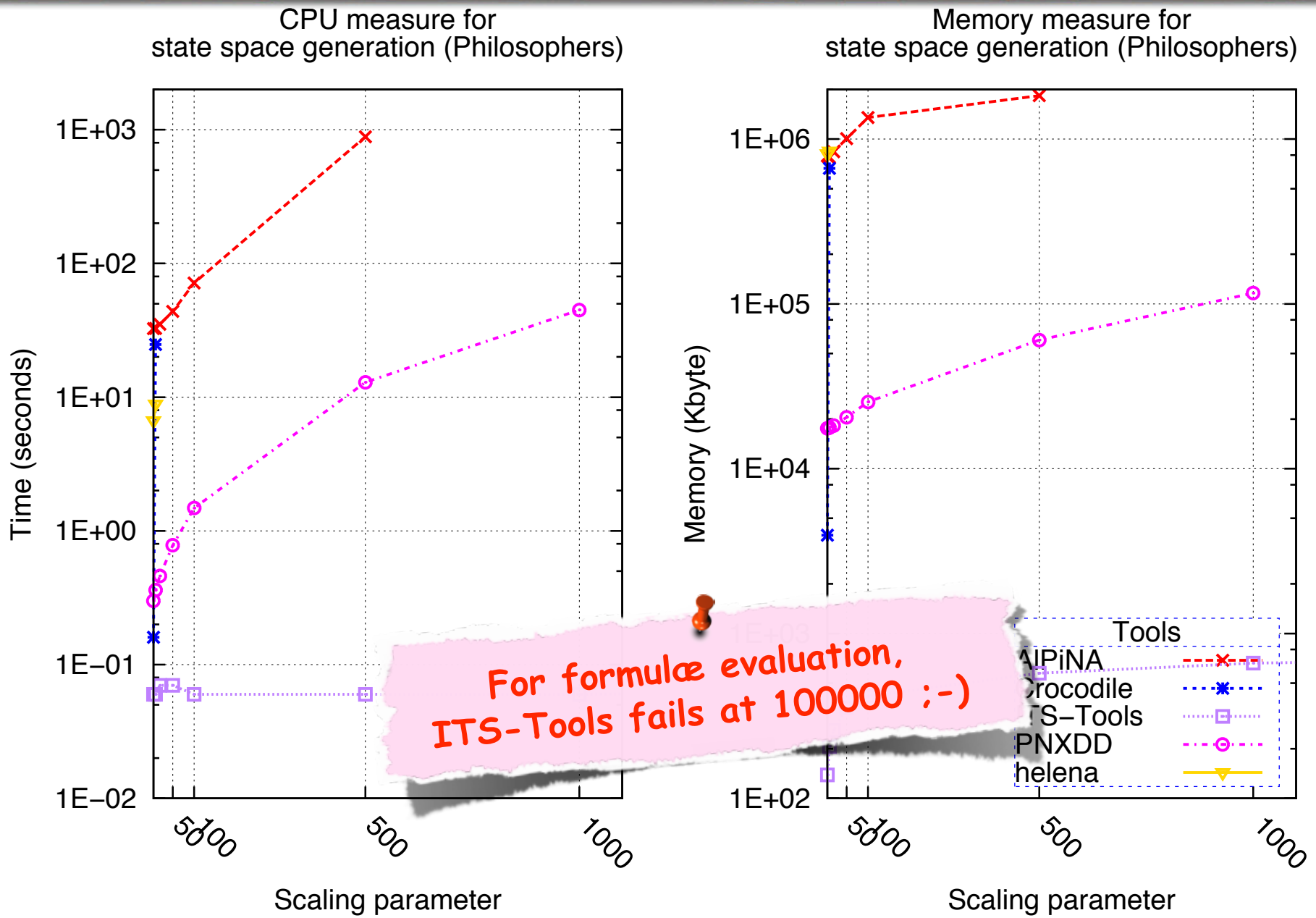
CPU measure for state space generation (Philosophers)



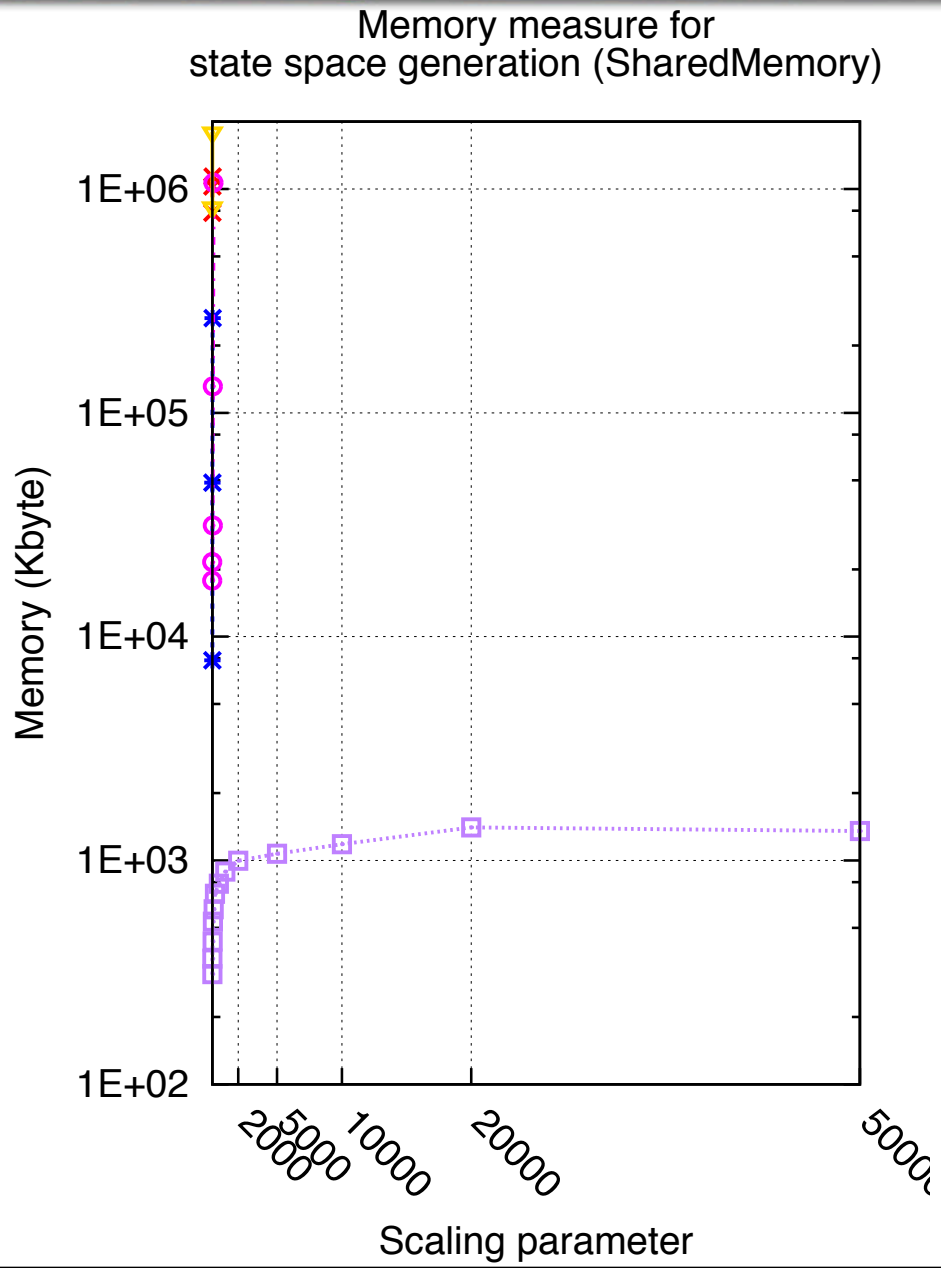
Memory measure for state space generation (Philosophers)

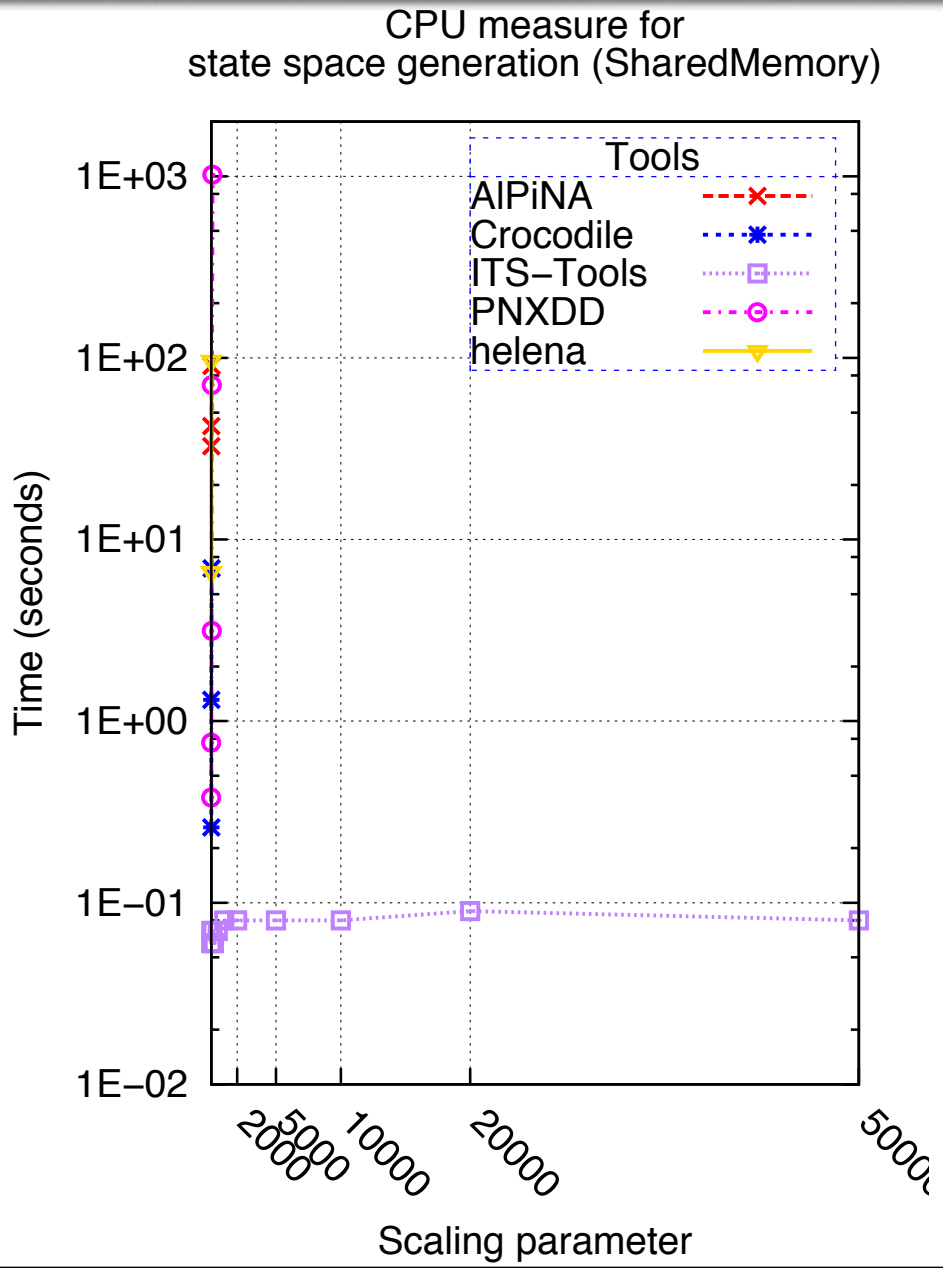


Philosopher, State Space Generation

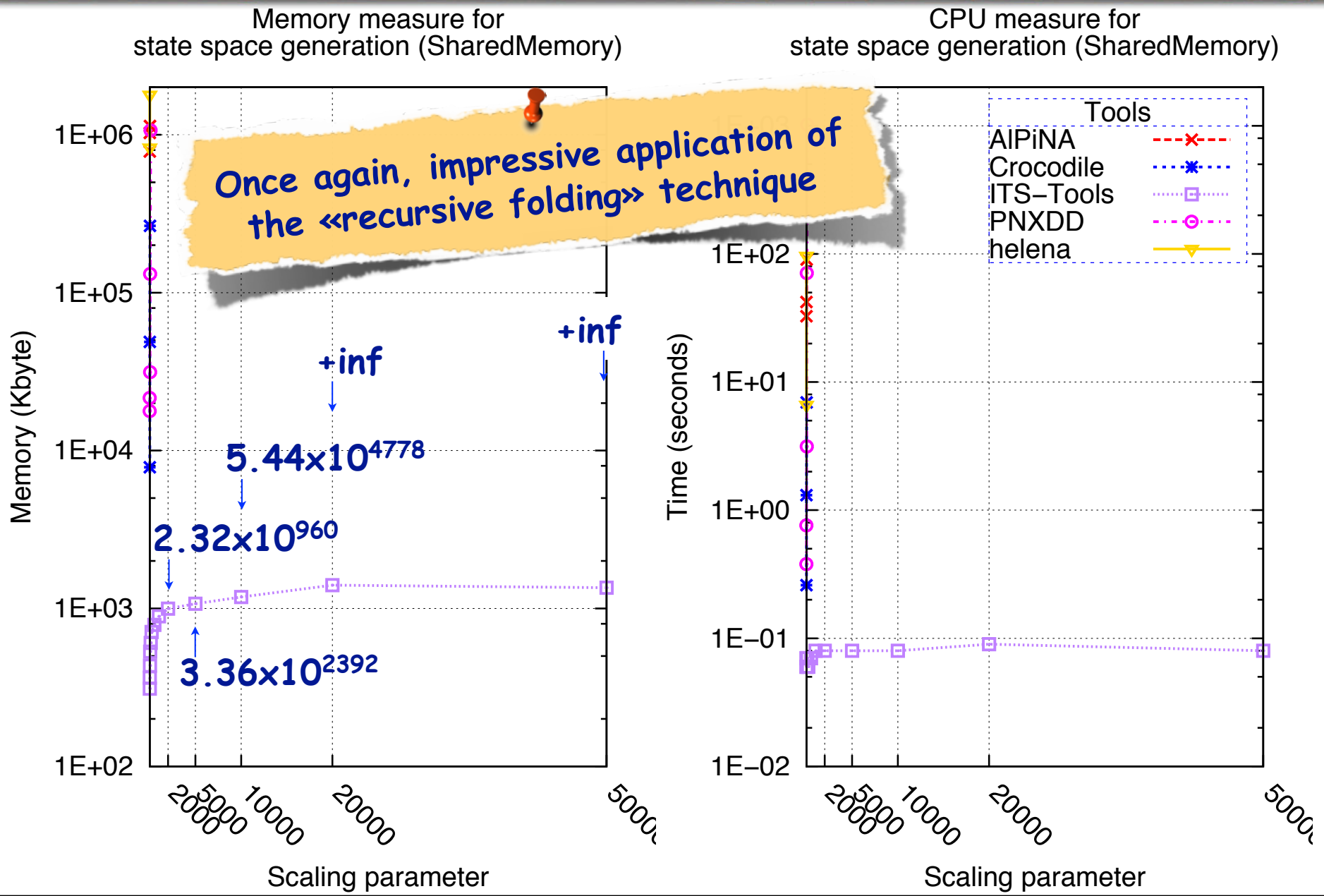


SharedMemory, State Space Generation



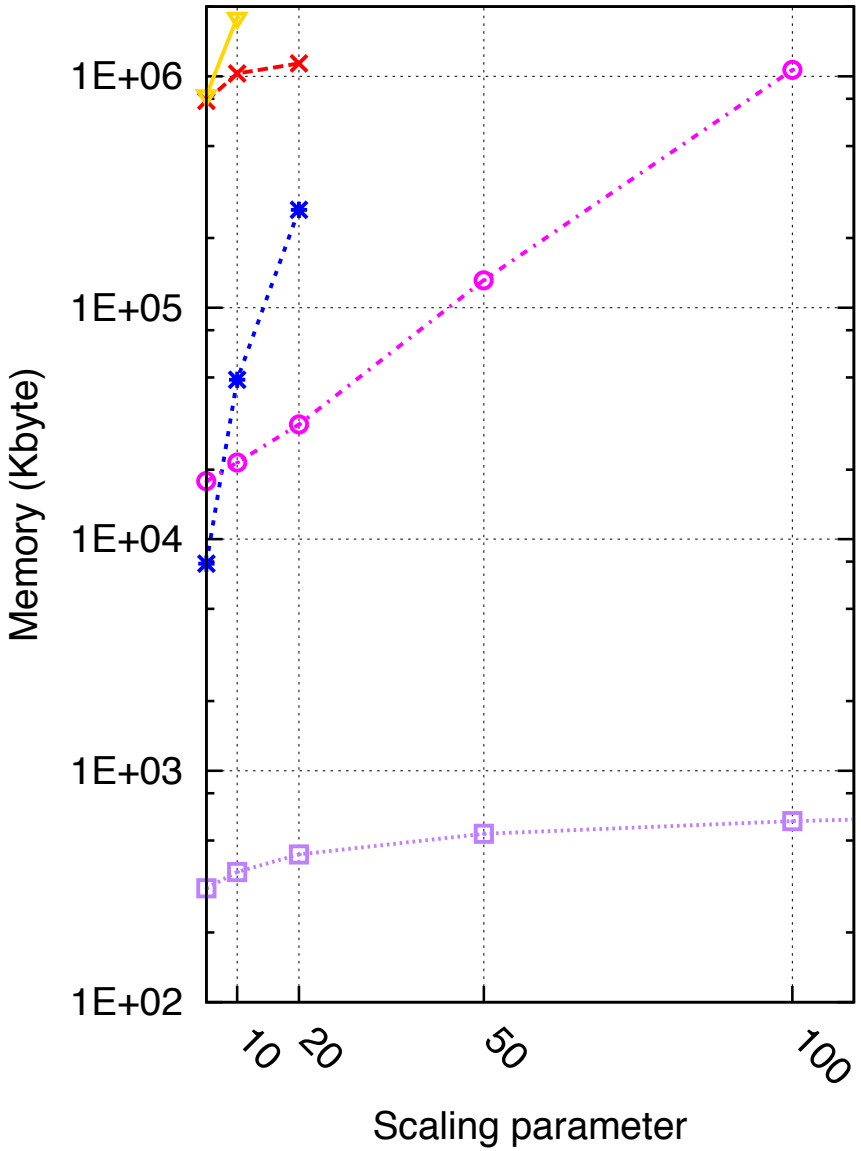


SharedMemory, State Space Generation

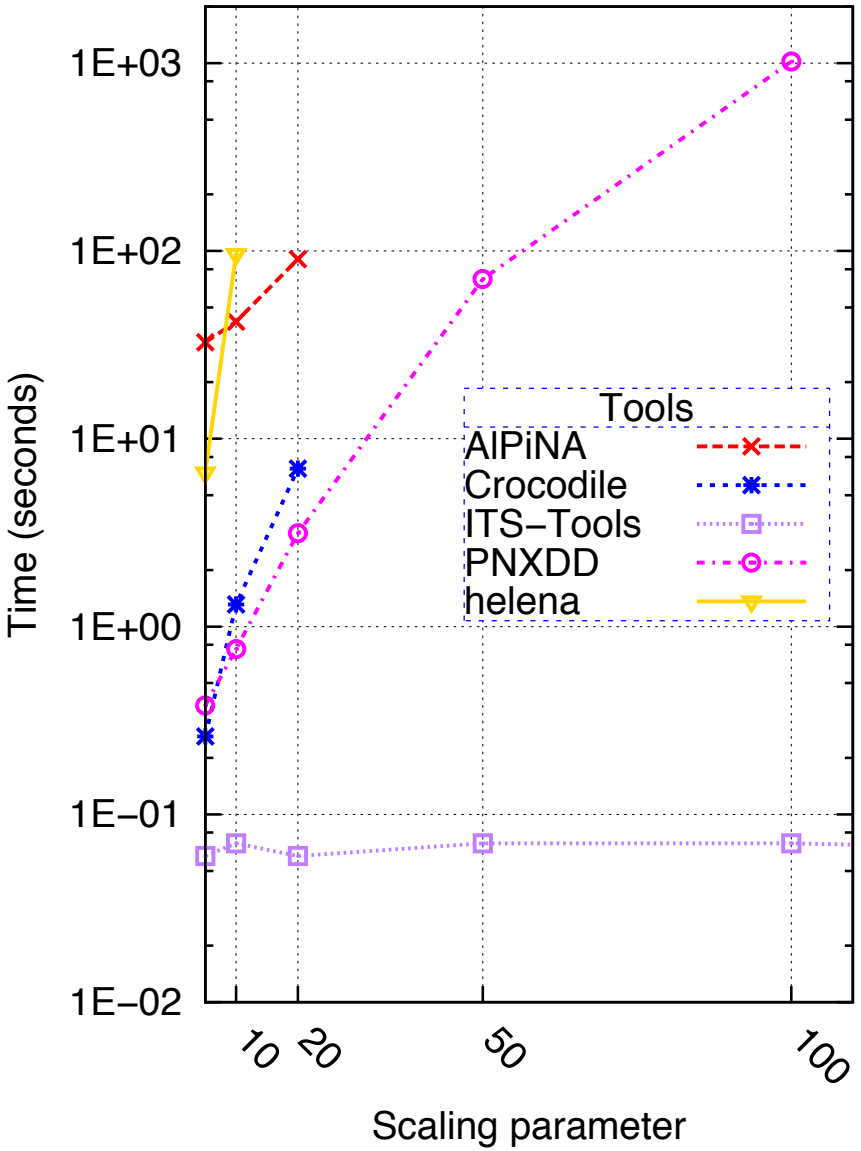


SharedMemory, State Space Generation

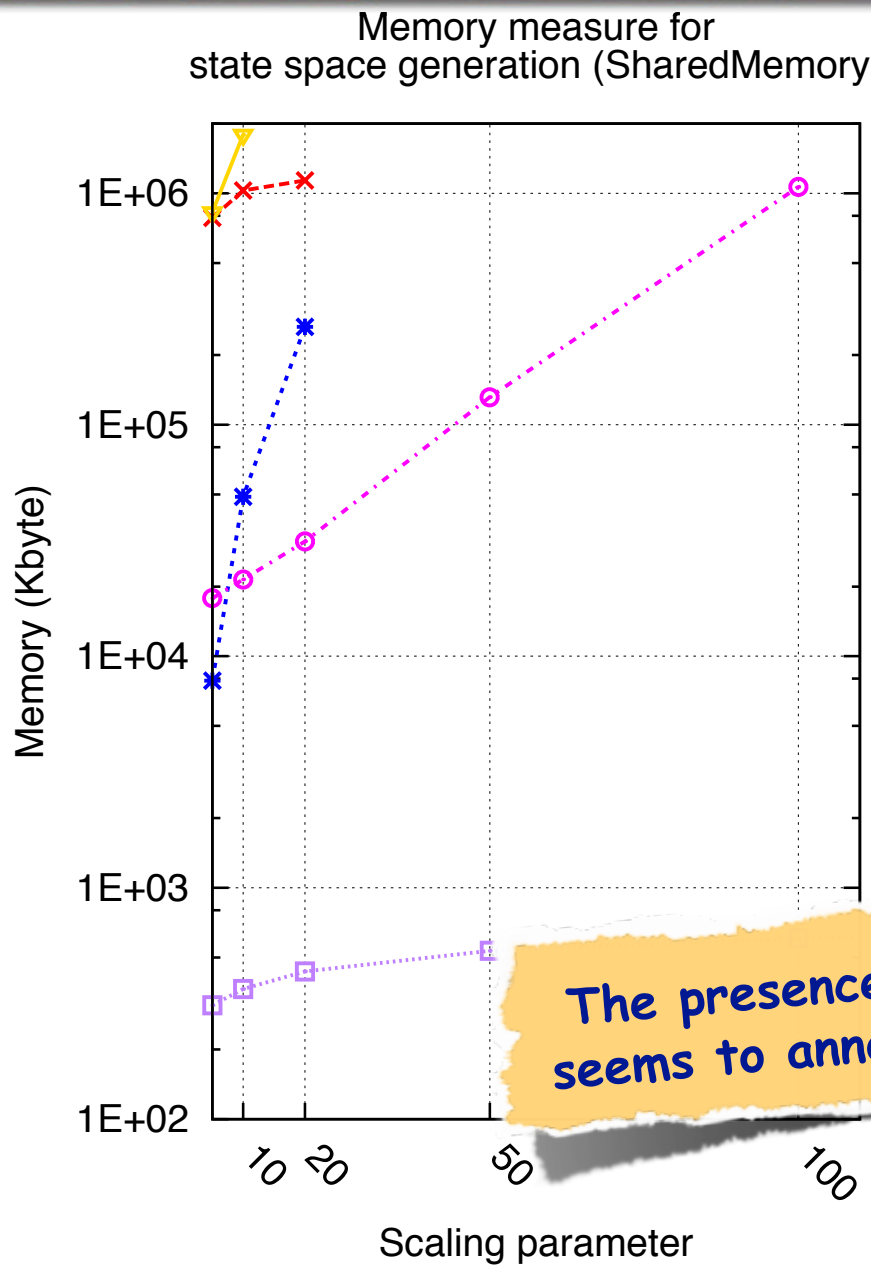
Memory measure for
state space generation (SharedMemory)

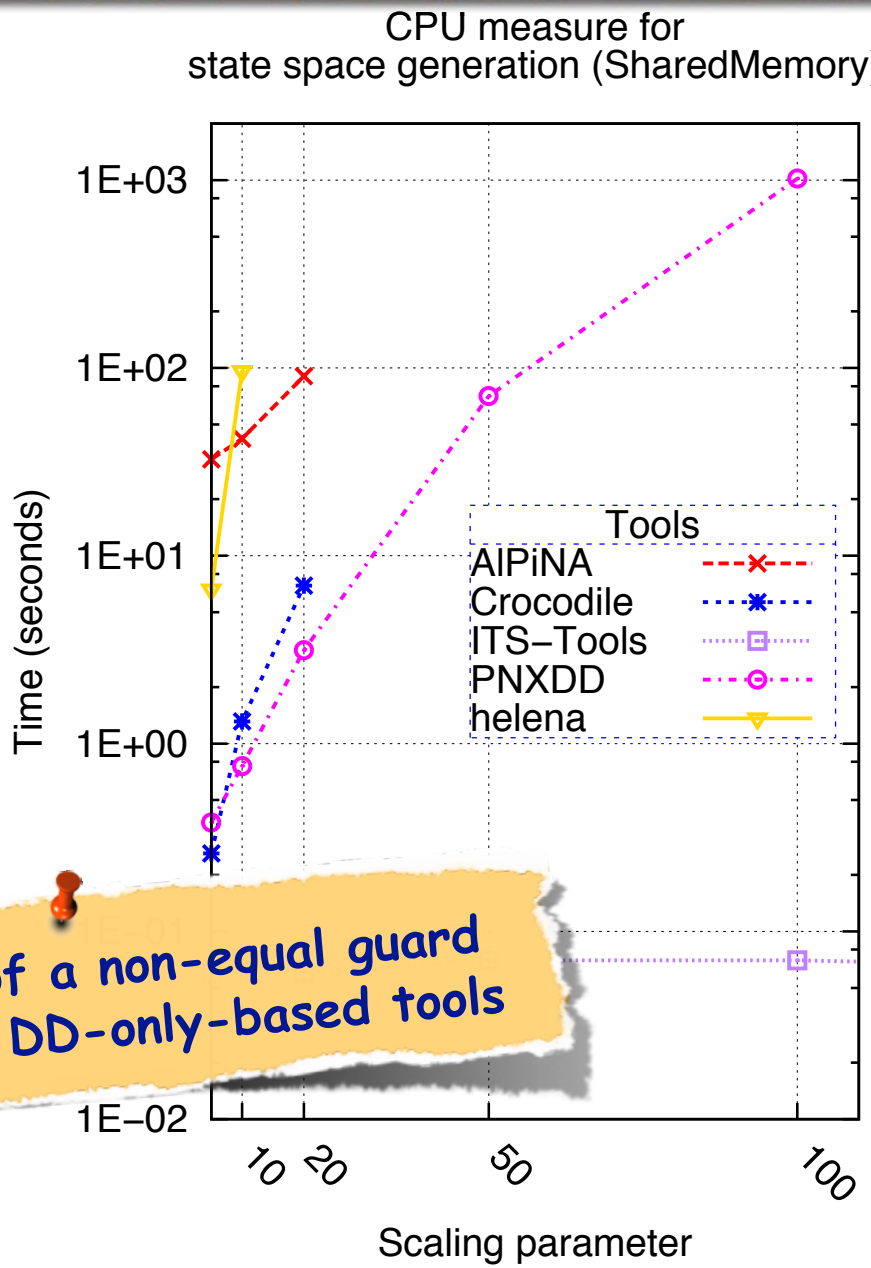


CPU measure for
state space generation (SharedMemory)



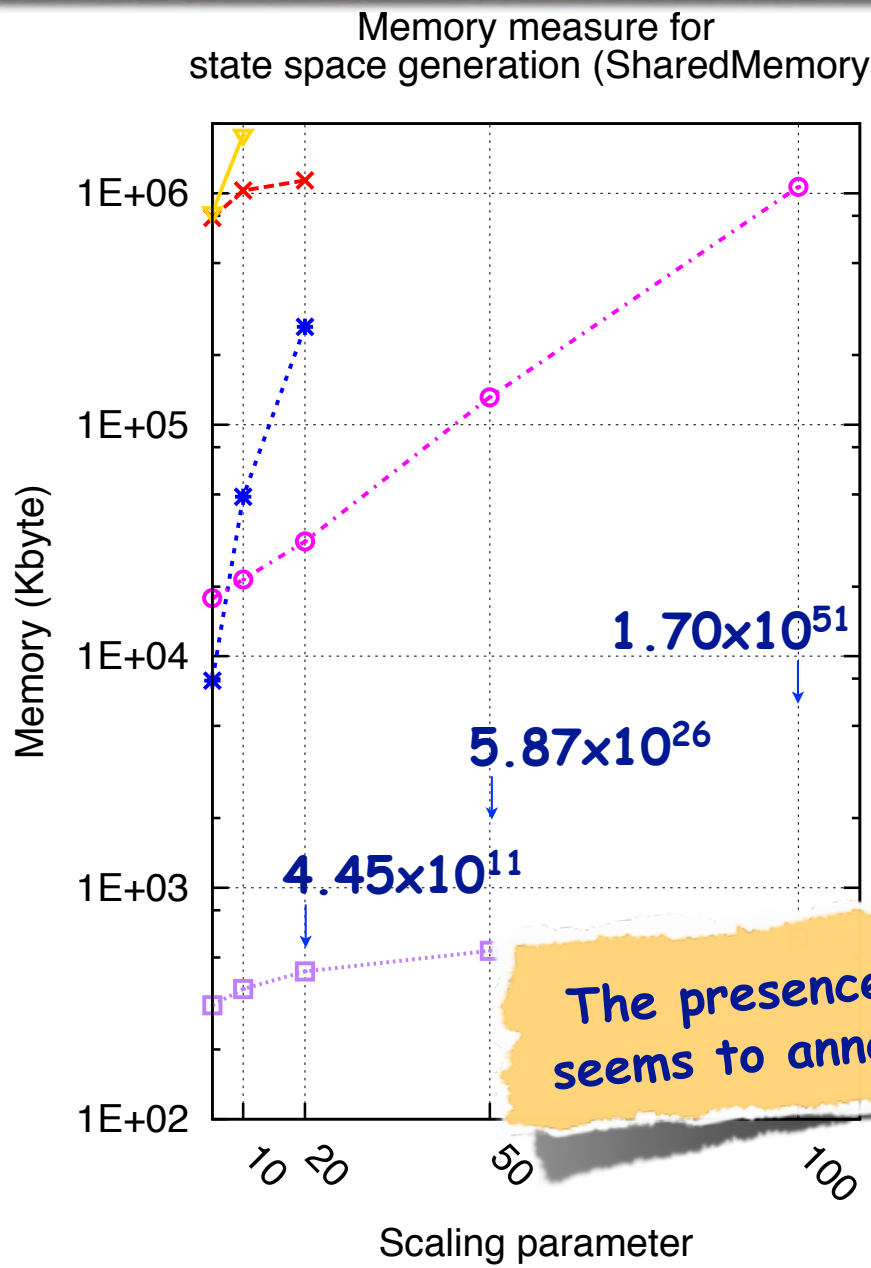
SharedMemory, State Space Generation

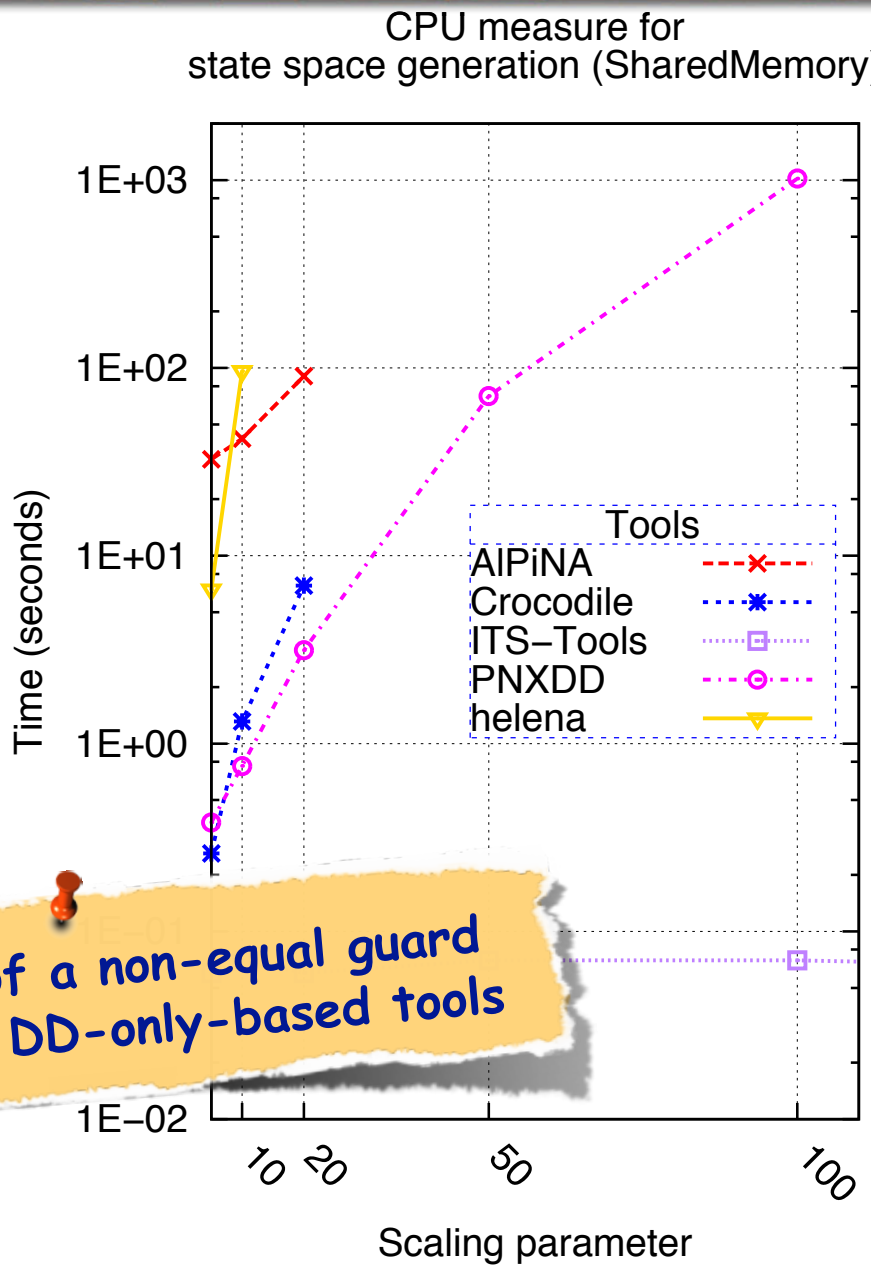




The presence of a non-equal guard seems to annoy DD-only-based tools

SharedMemory, State Space Generation

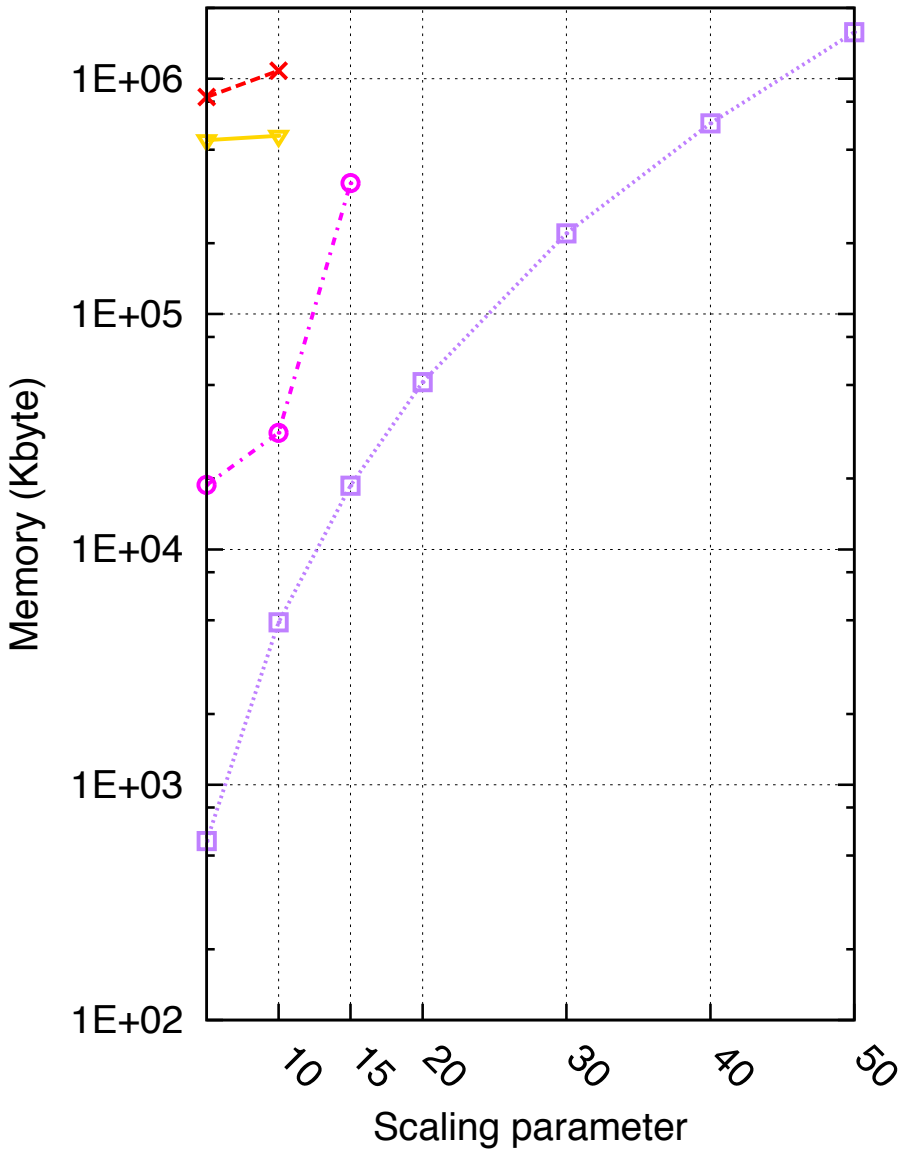




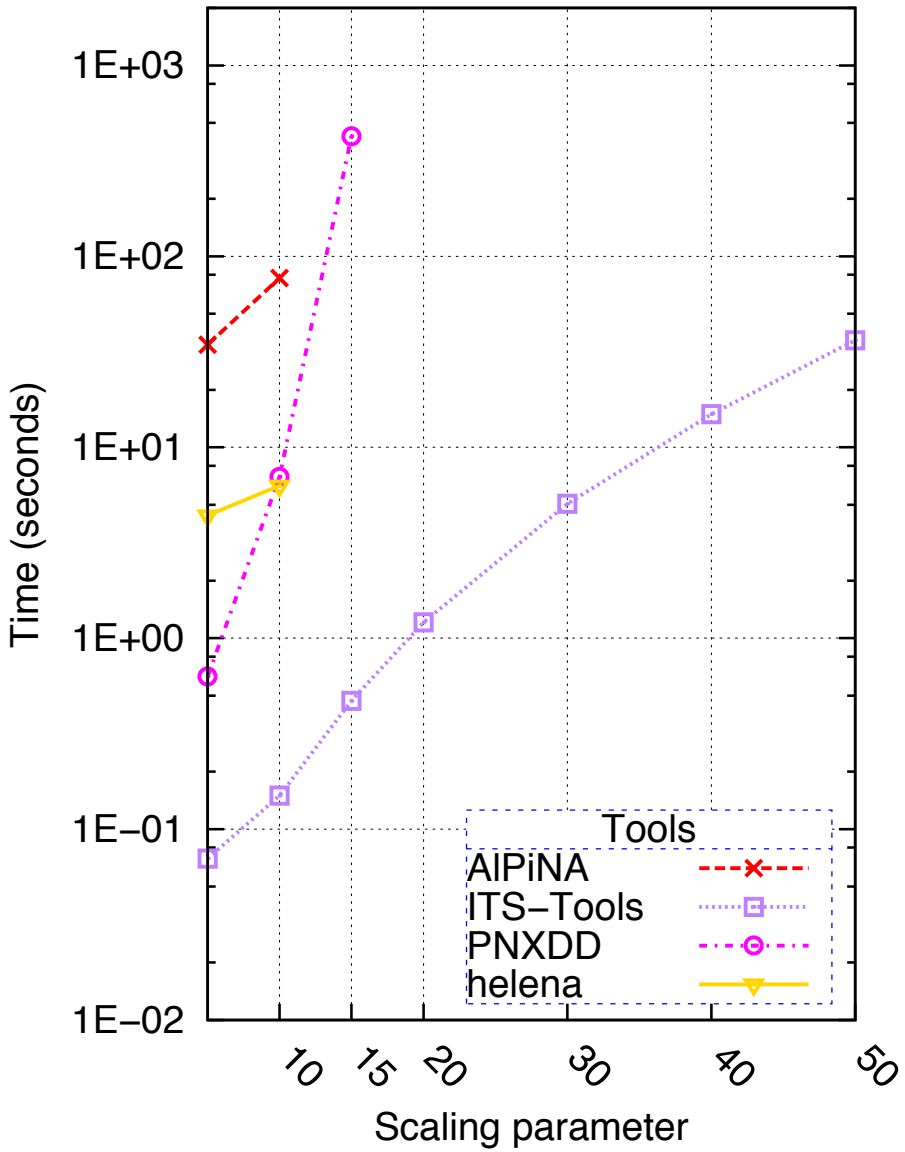
The presence of a non-equal guard seems to annoy DD-only-based tools

TokenRing, State Space Generation

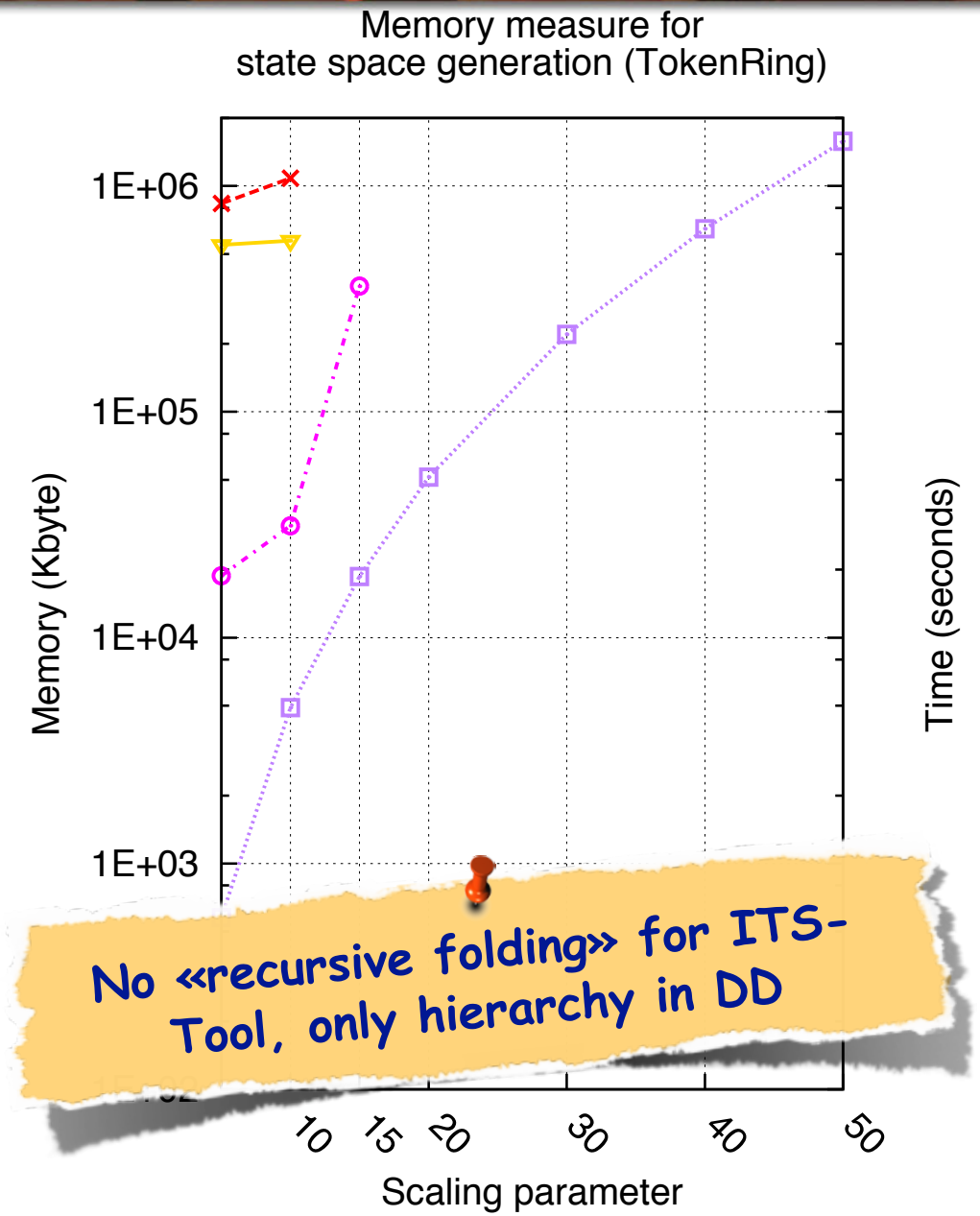
Memory measure for state space generation (TokenRing)



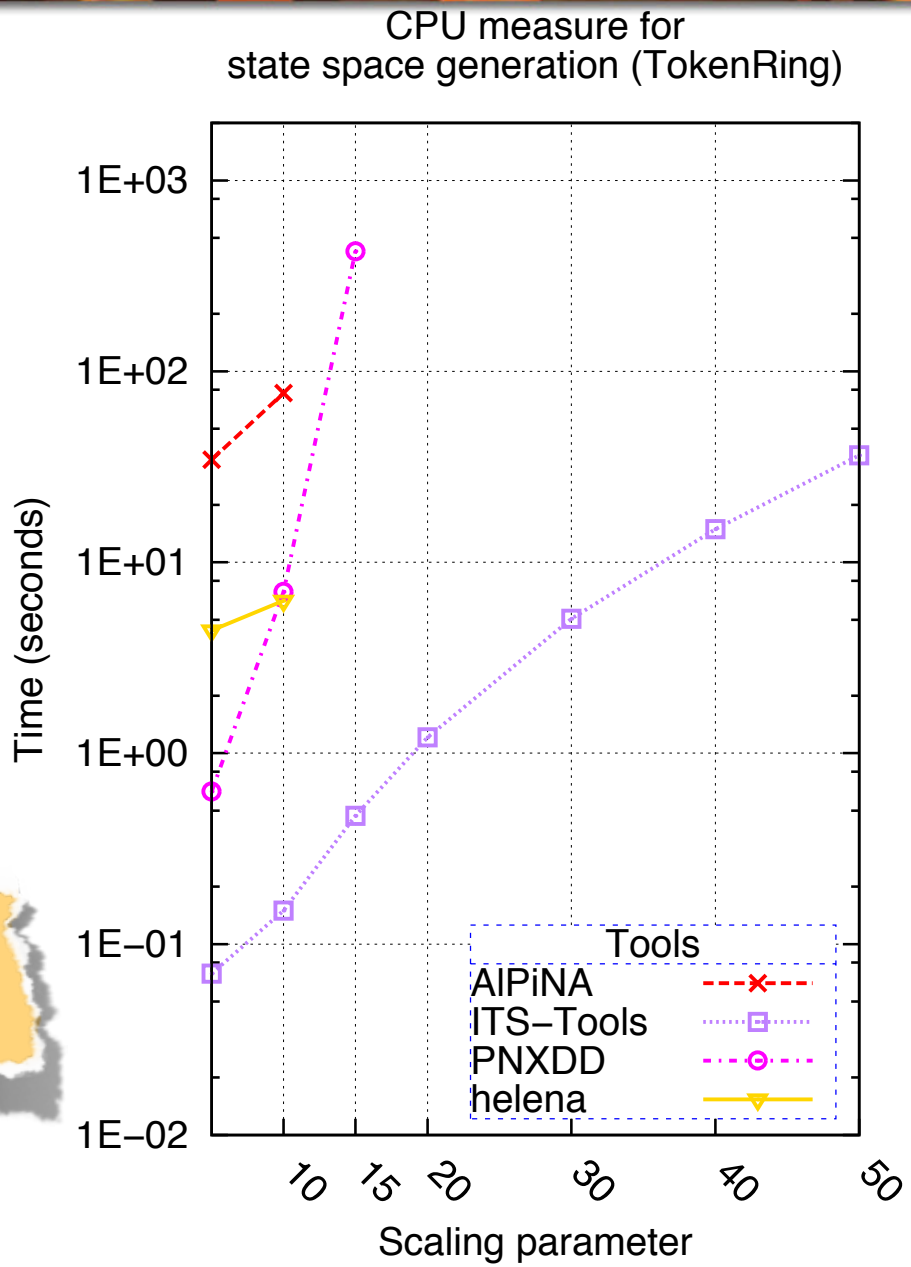
CPU measure for state space generation (TokenRing)



TokenRing, State Space Generation

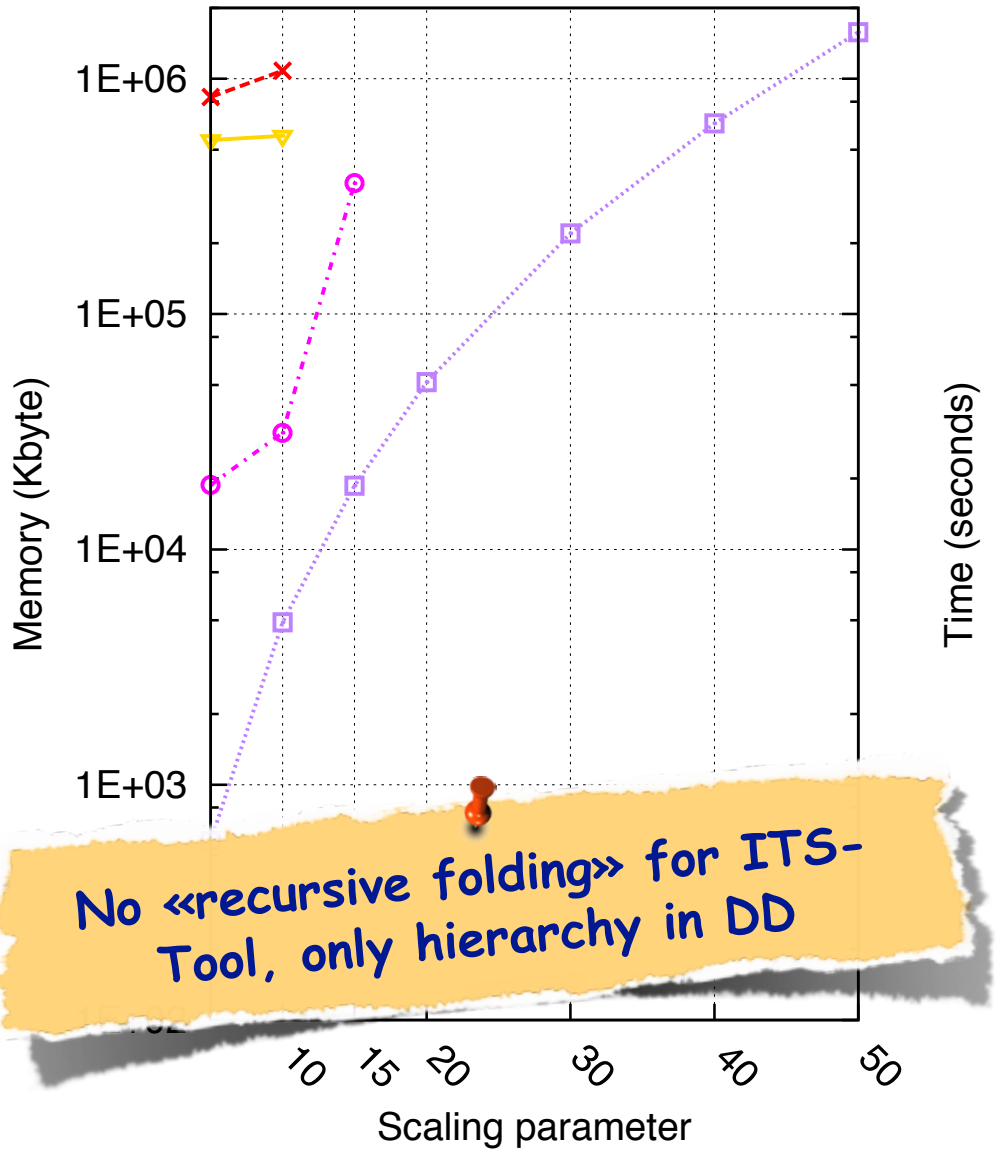


No «recursive folding» for ITS-Tool, only hierarchy in DD

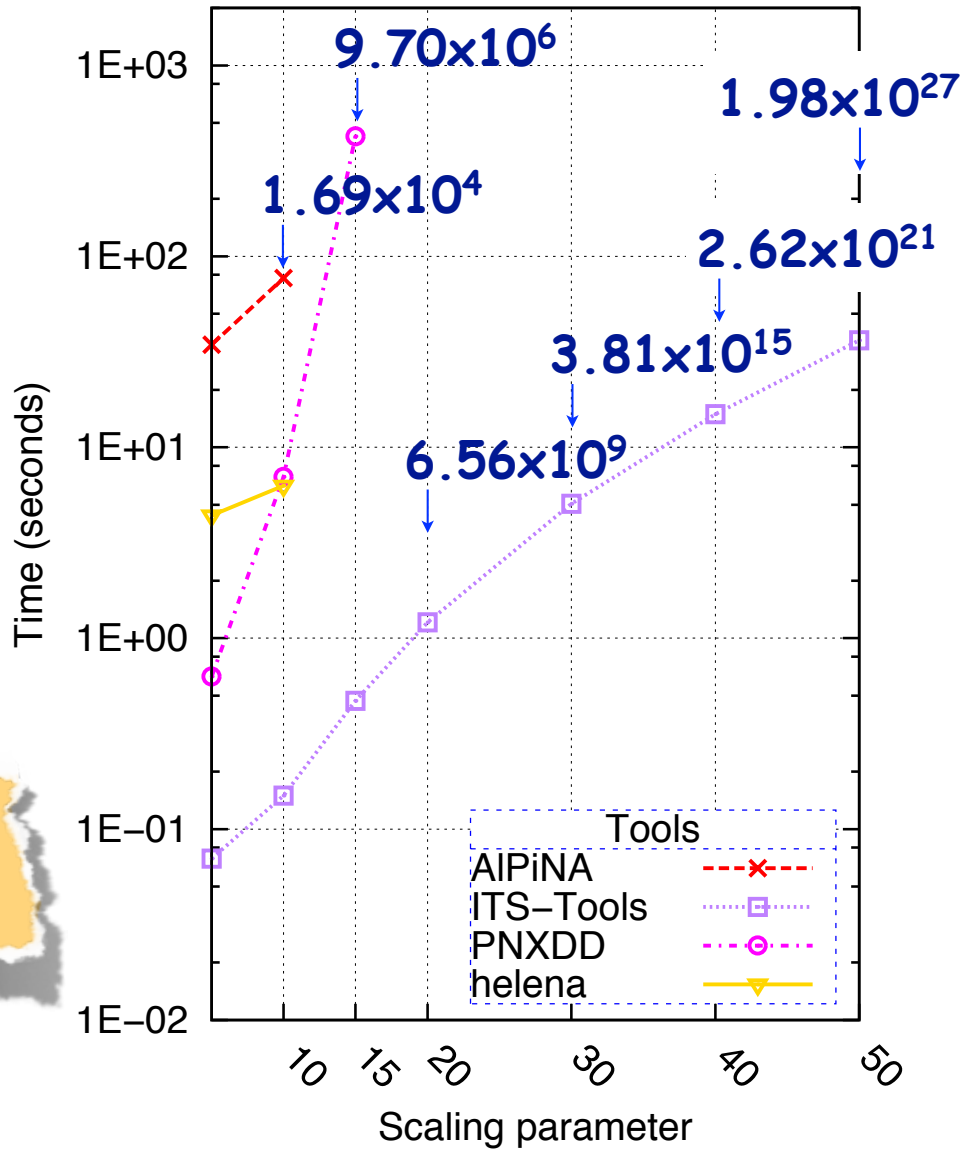


TokenRing, State Space Generation

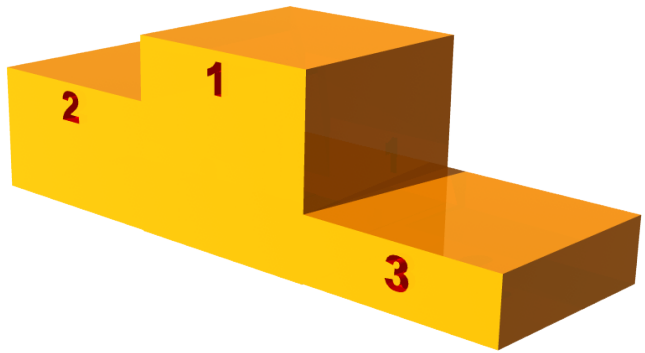
Memory measure for state space generation (TokenRing)



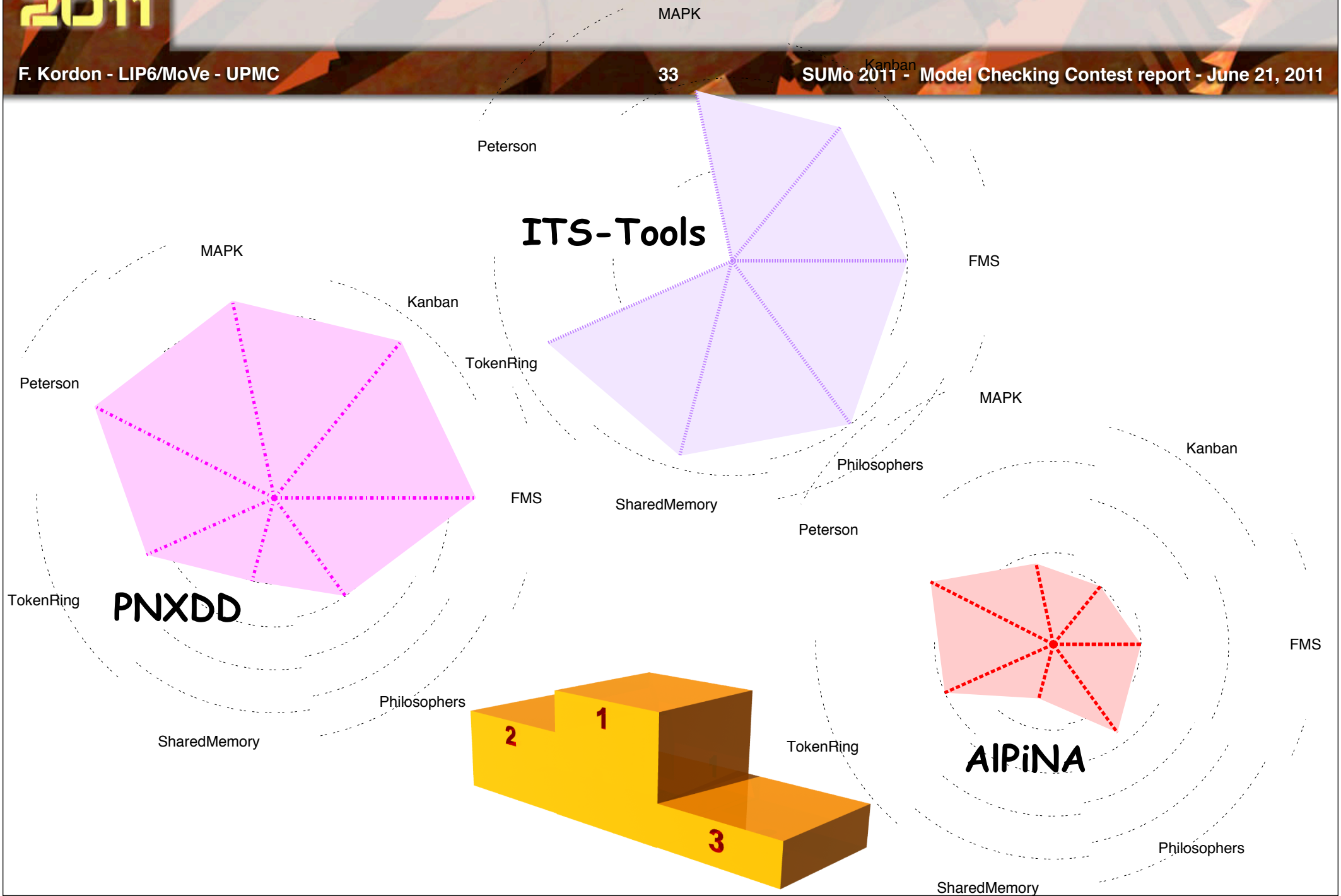
CPU measure for state space generation (TokenRing)



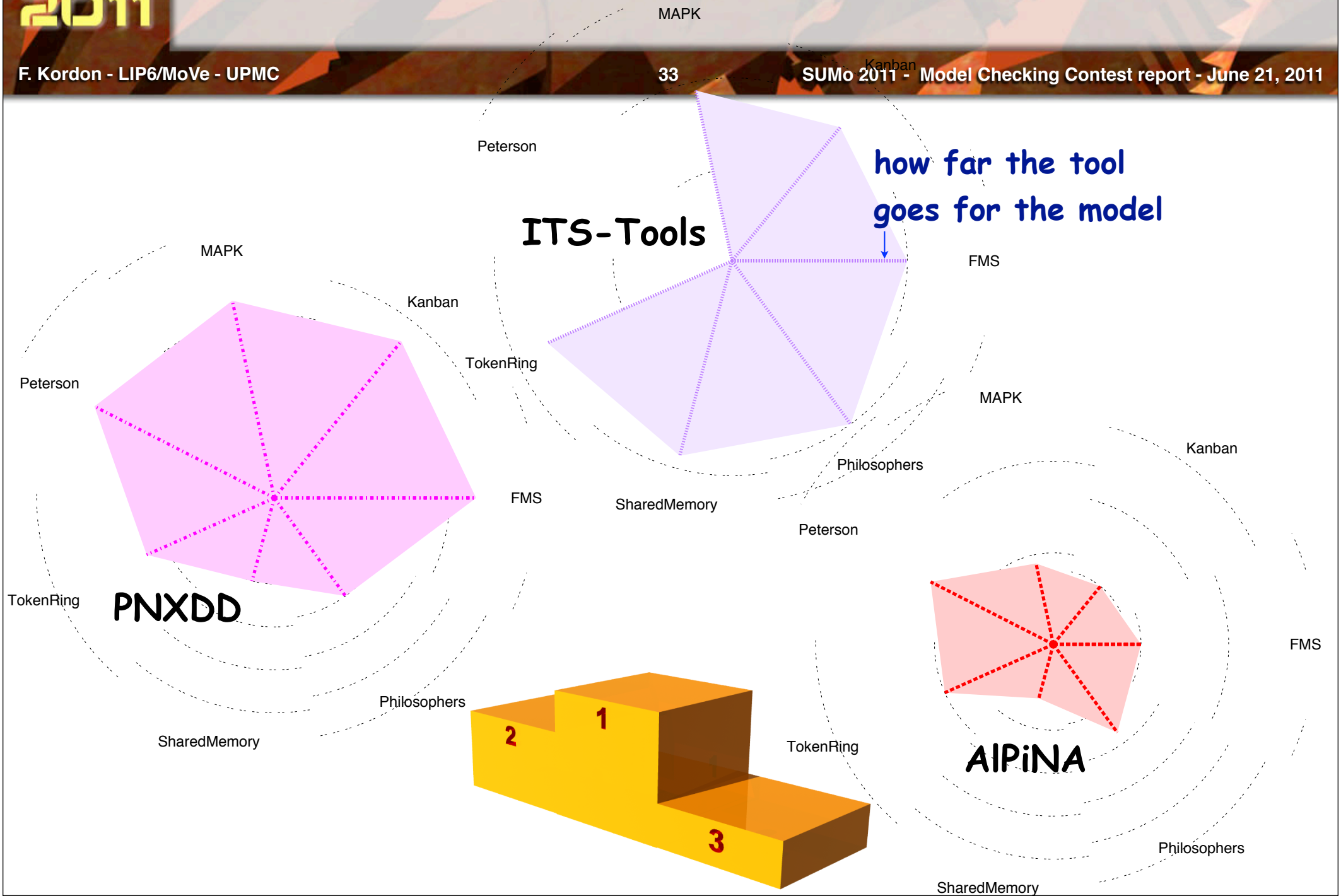
State Space Generation, Best Tools



State Space Generation, Best Tools

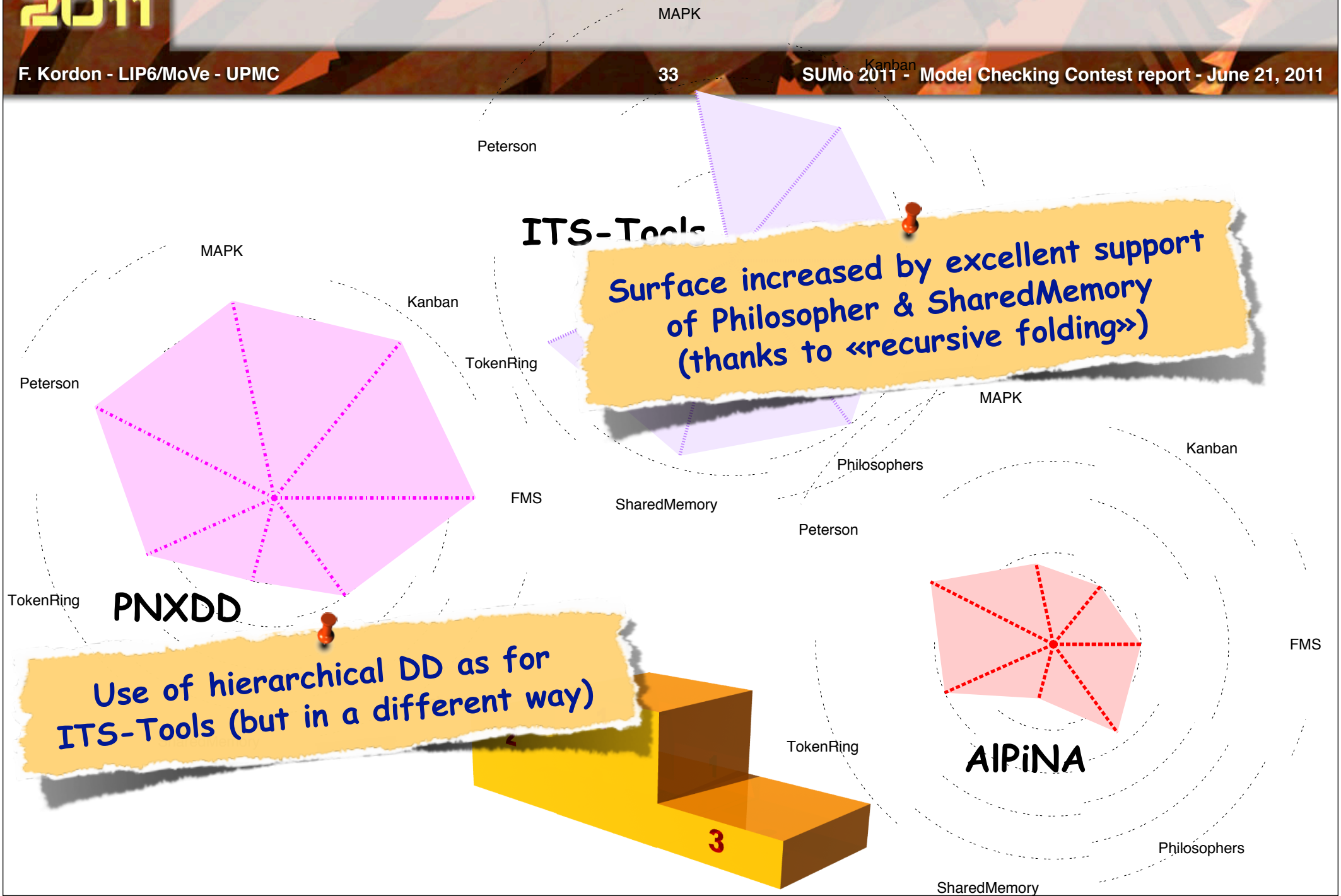


State Space Generation, Best Tools

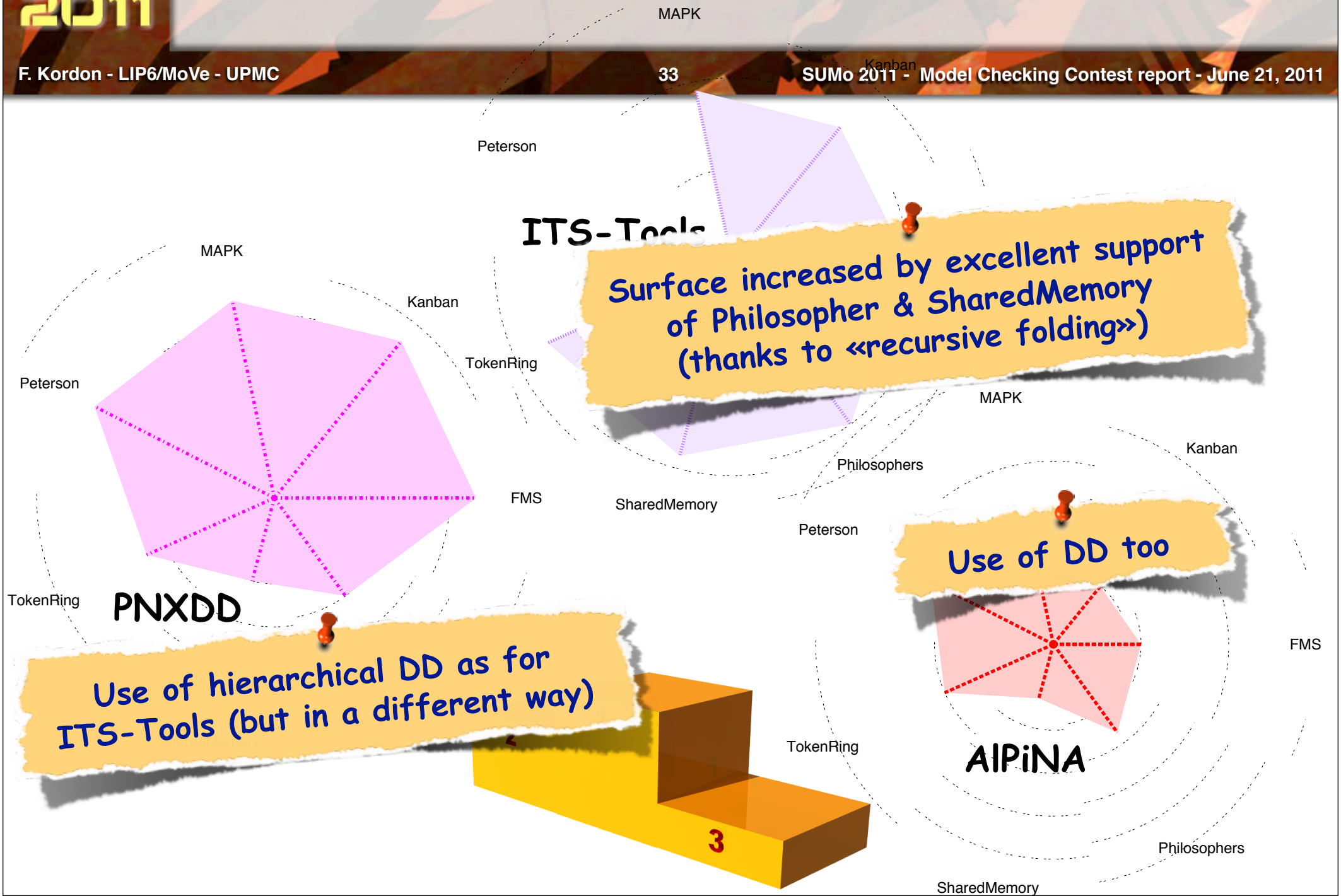


State Space Generation, Best Tools

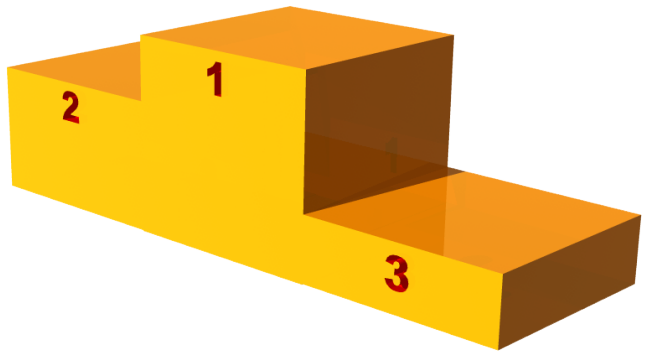
State Space Generation, Best Tools



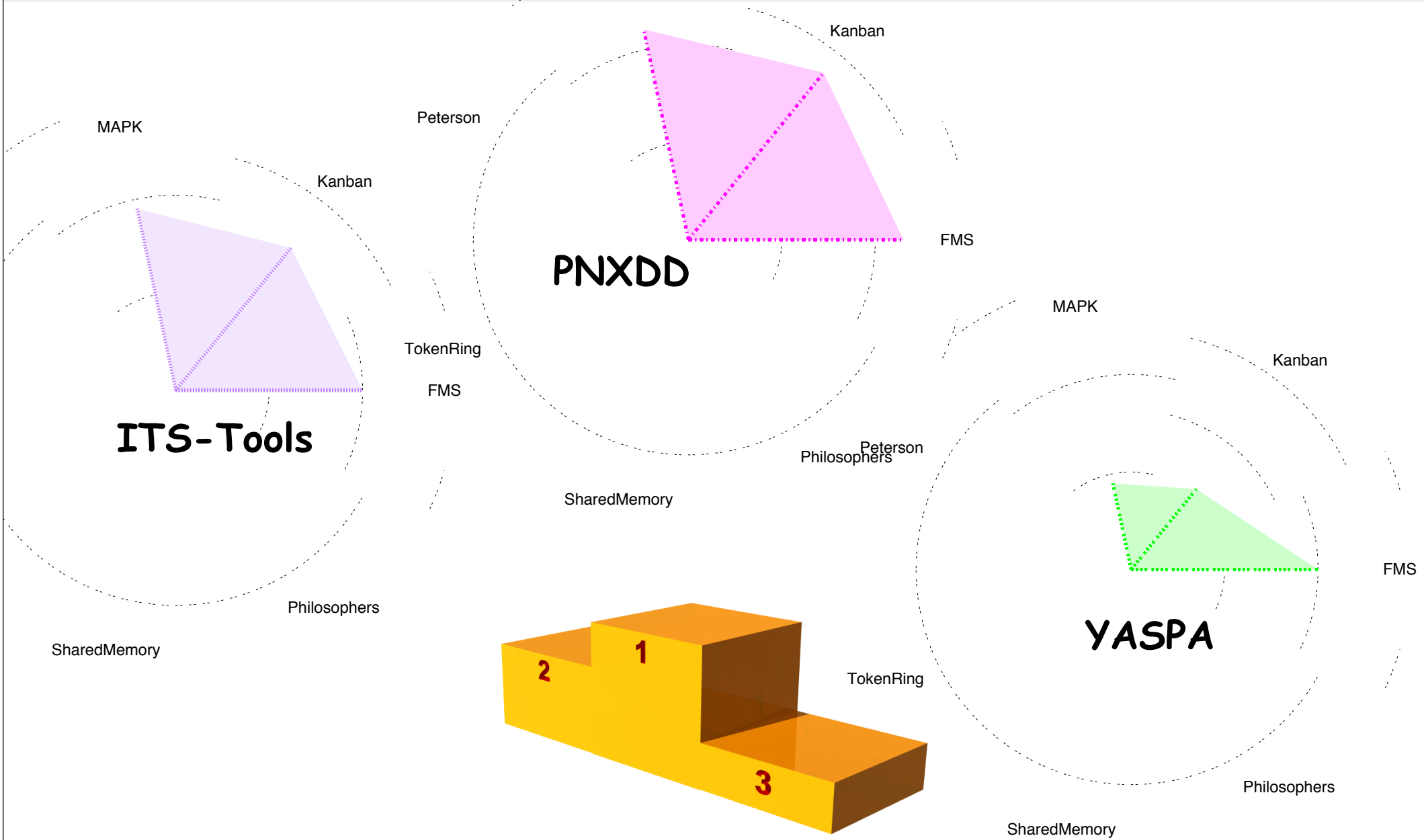
State Space Generation, Best Tools



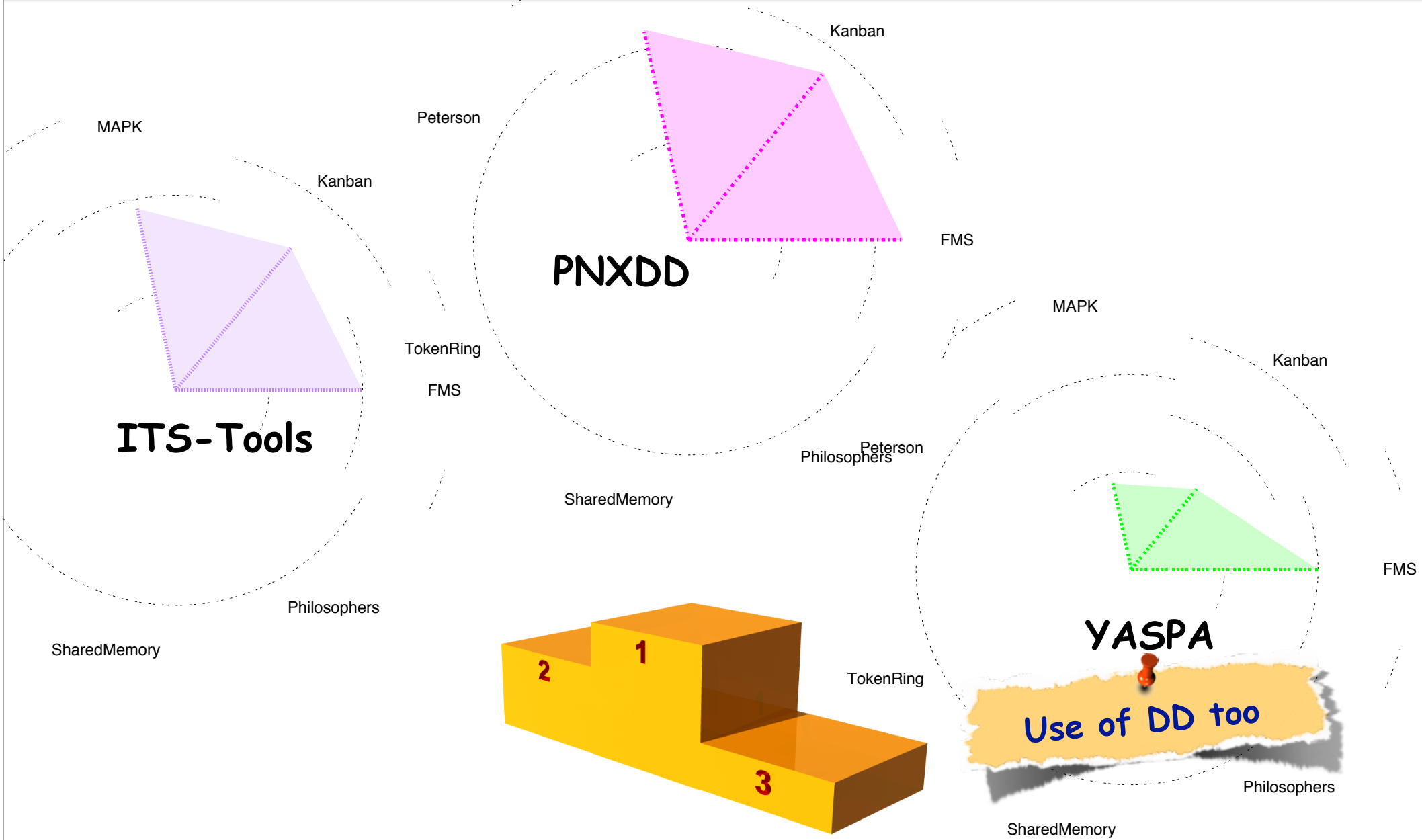
State Space Generation, Best Tools (P/T Nets)



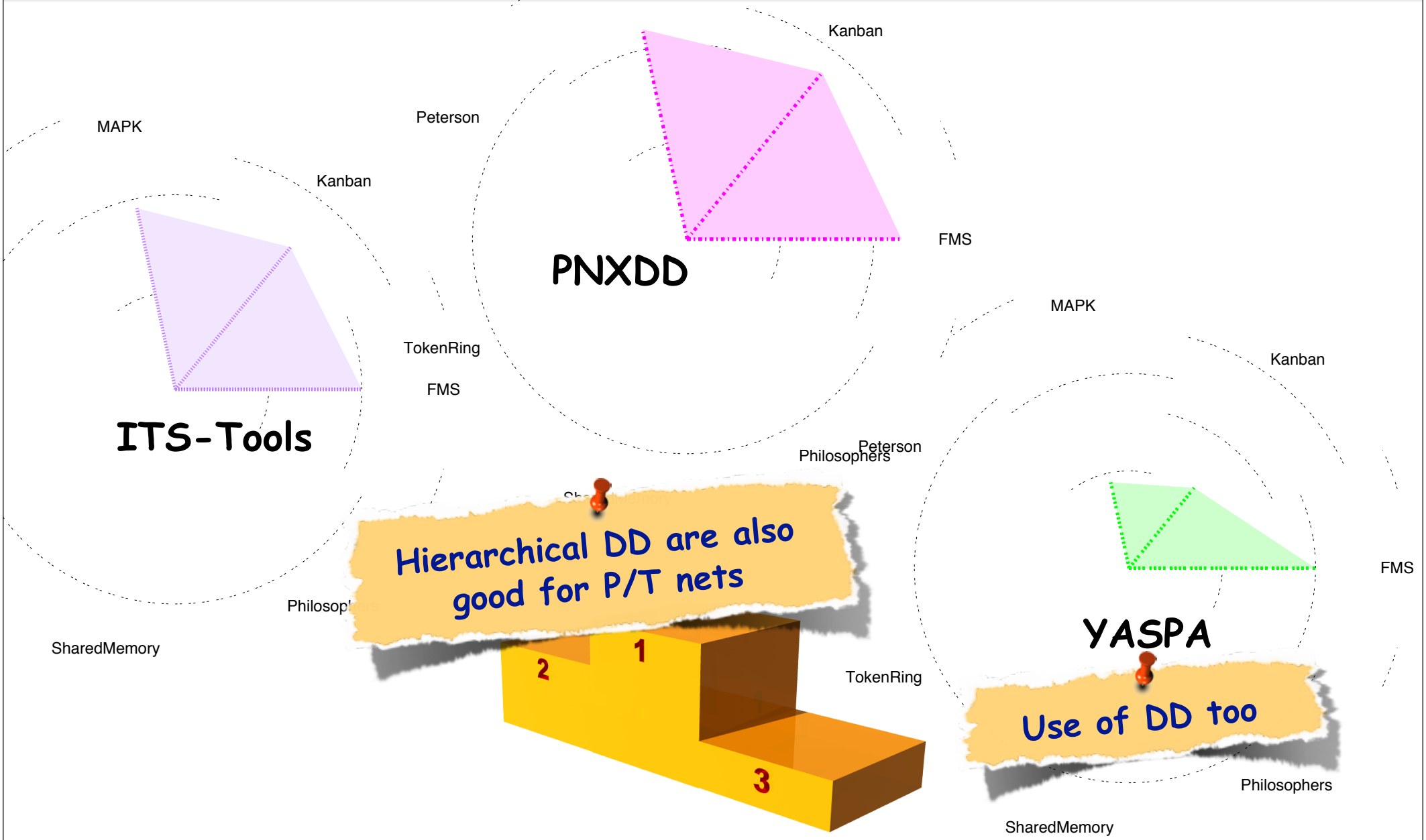
State Space Generation, Best Tools (P/T Nets)



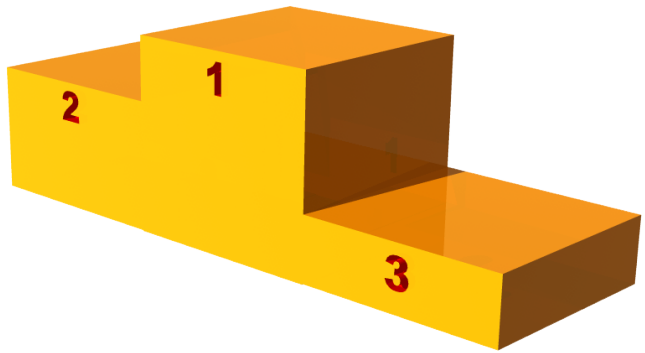
State Space Generation, Best Tools (P/T Nets)



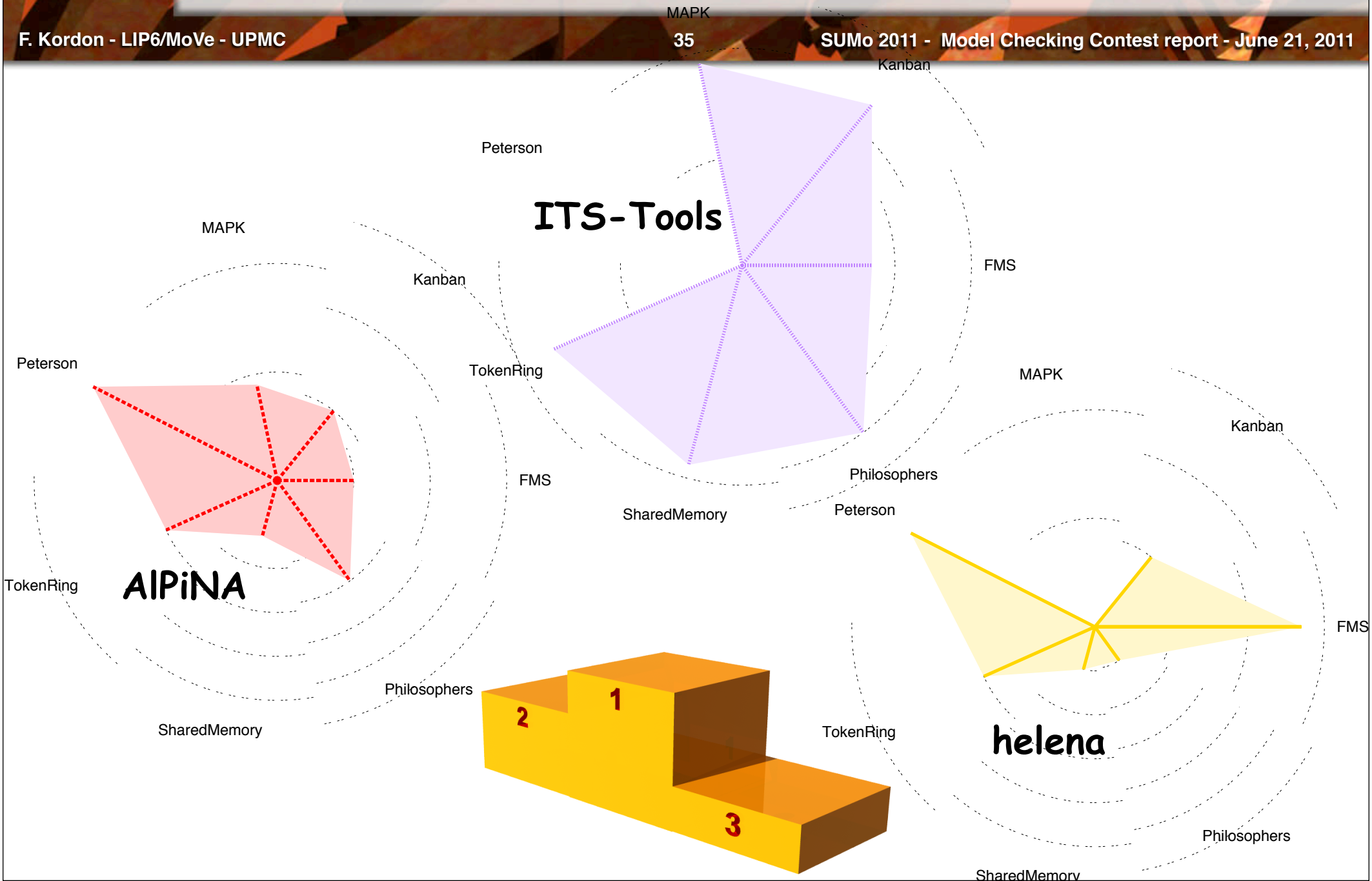
State Space Generation, Best Tools (P/T Nets)



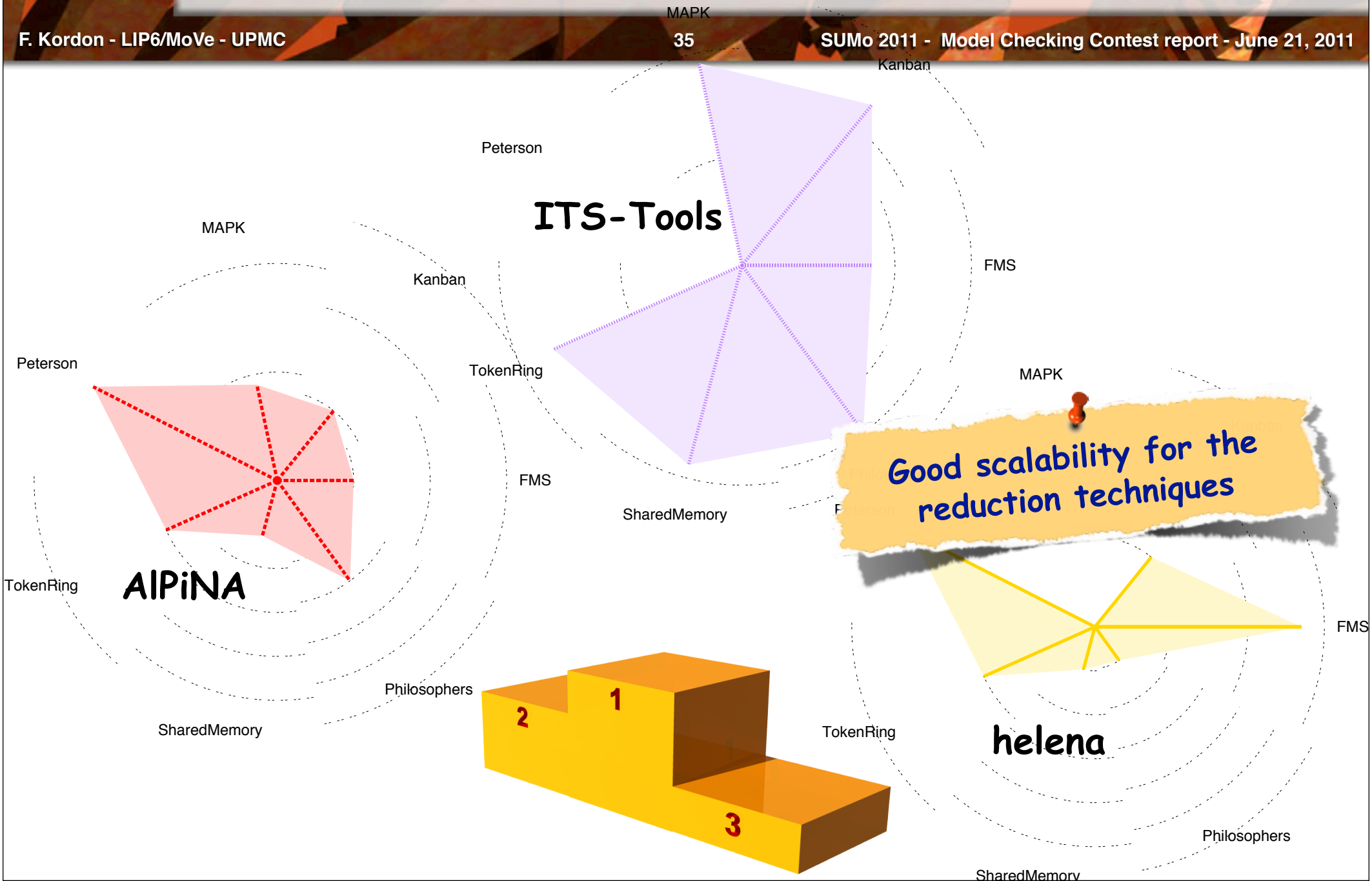
Deadlock Detection, Best Tools



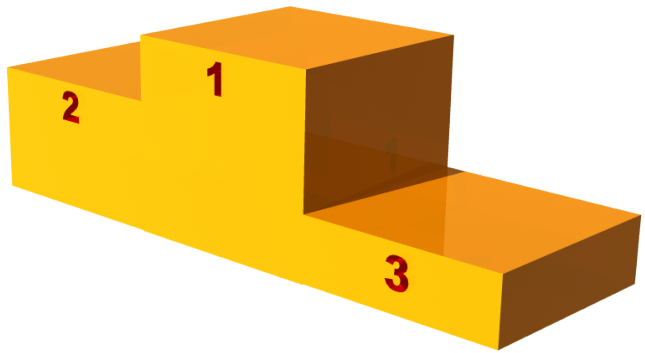
Deadlock Detection, Best Tools



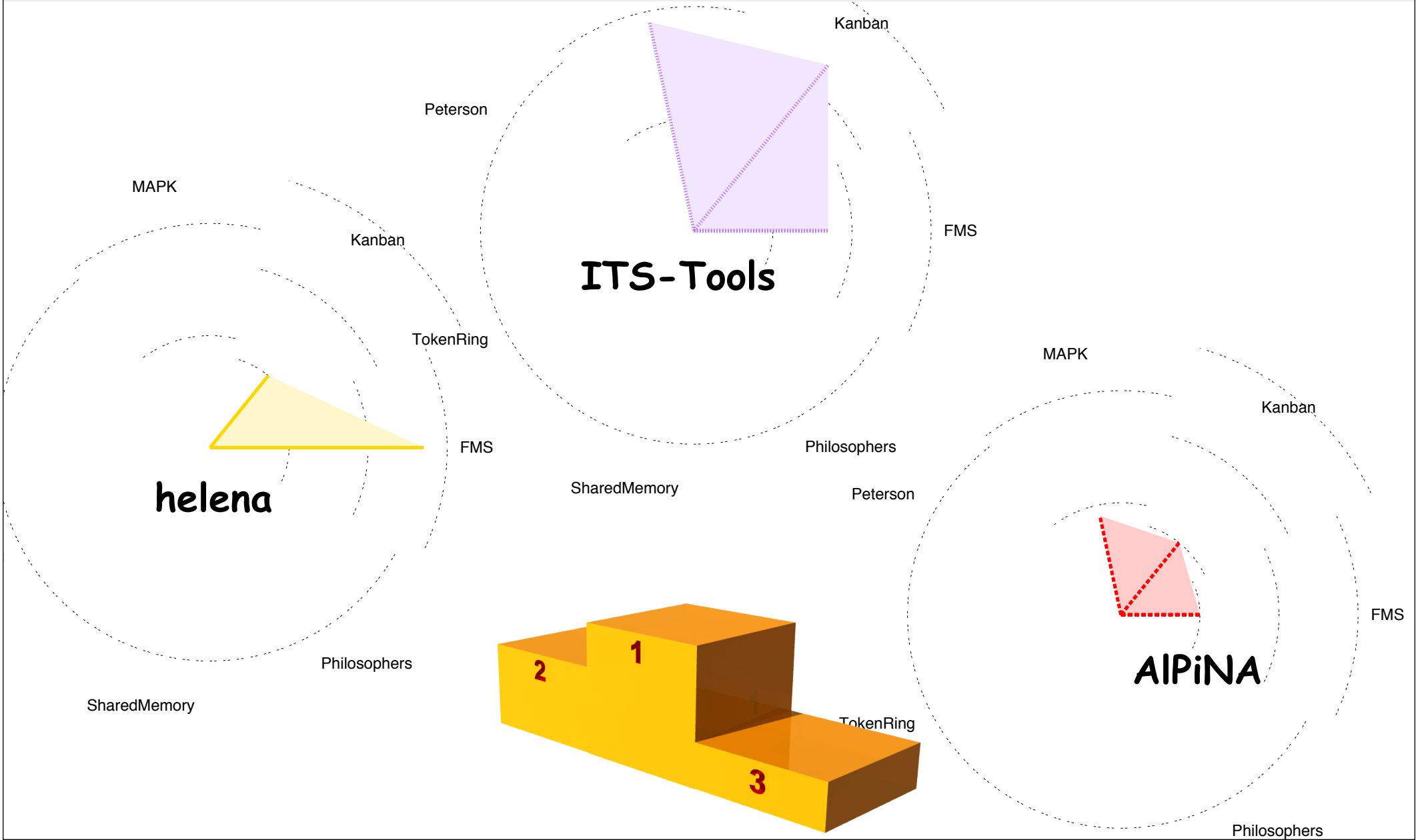
Deadlock Detection, Best Tools



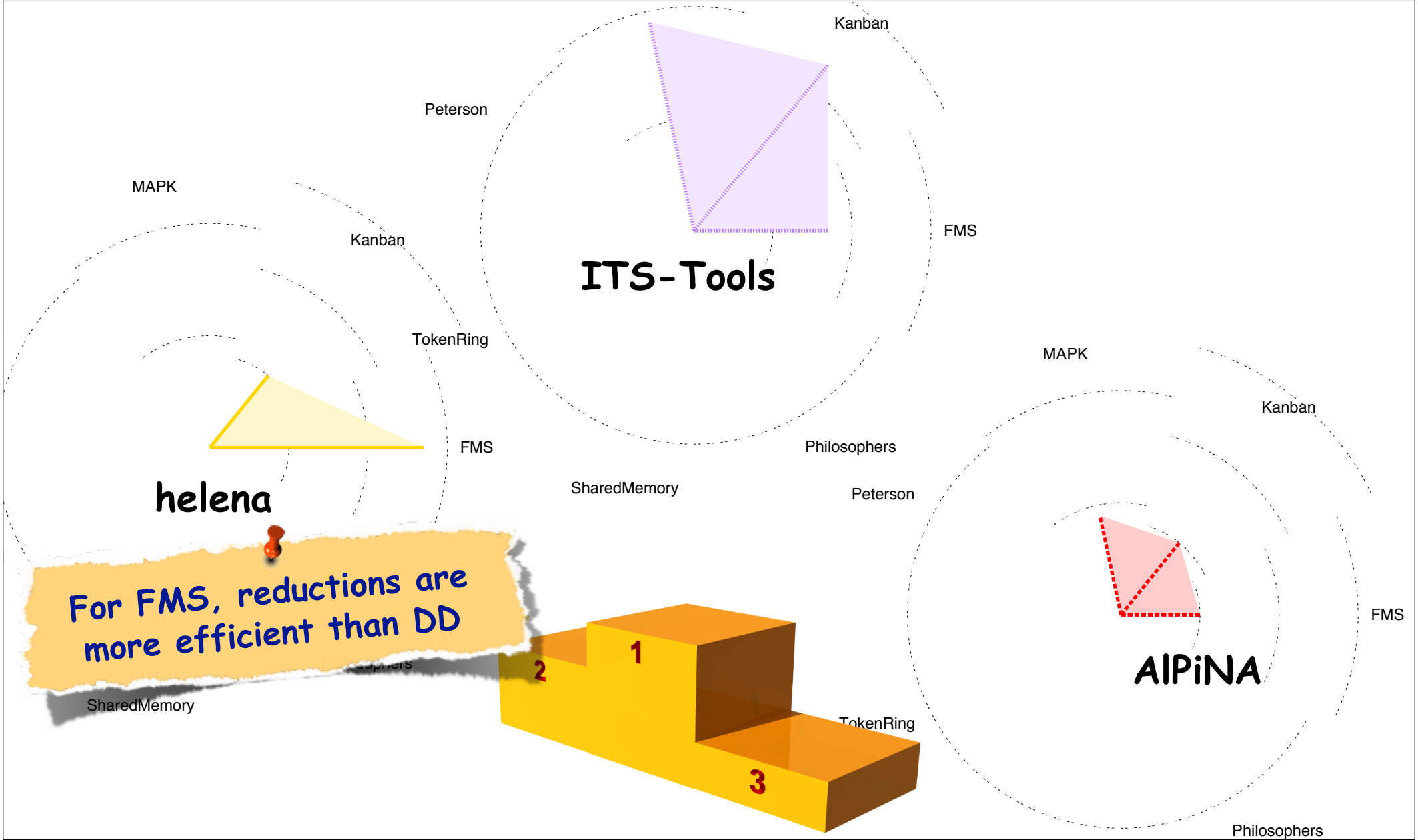
Deadlock Detection, Best Tools (P/T Nets)



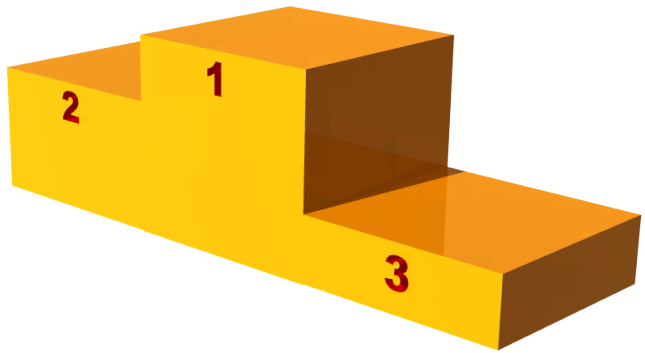
Deadlock Detection, Best Tools (P/T Nets)



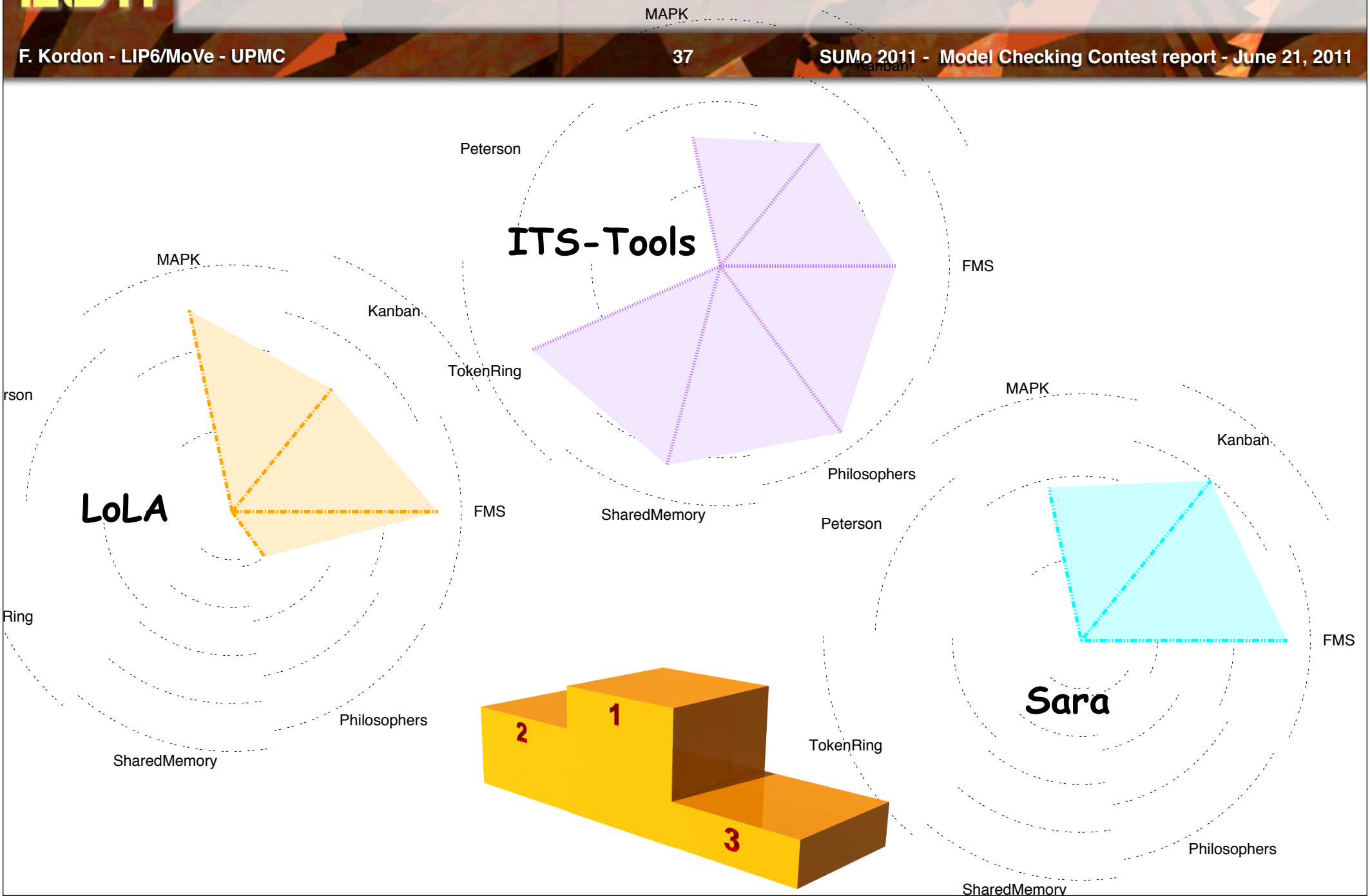
Deadlock Detection, Best Tools (P/T Nets)



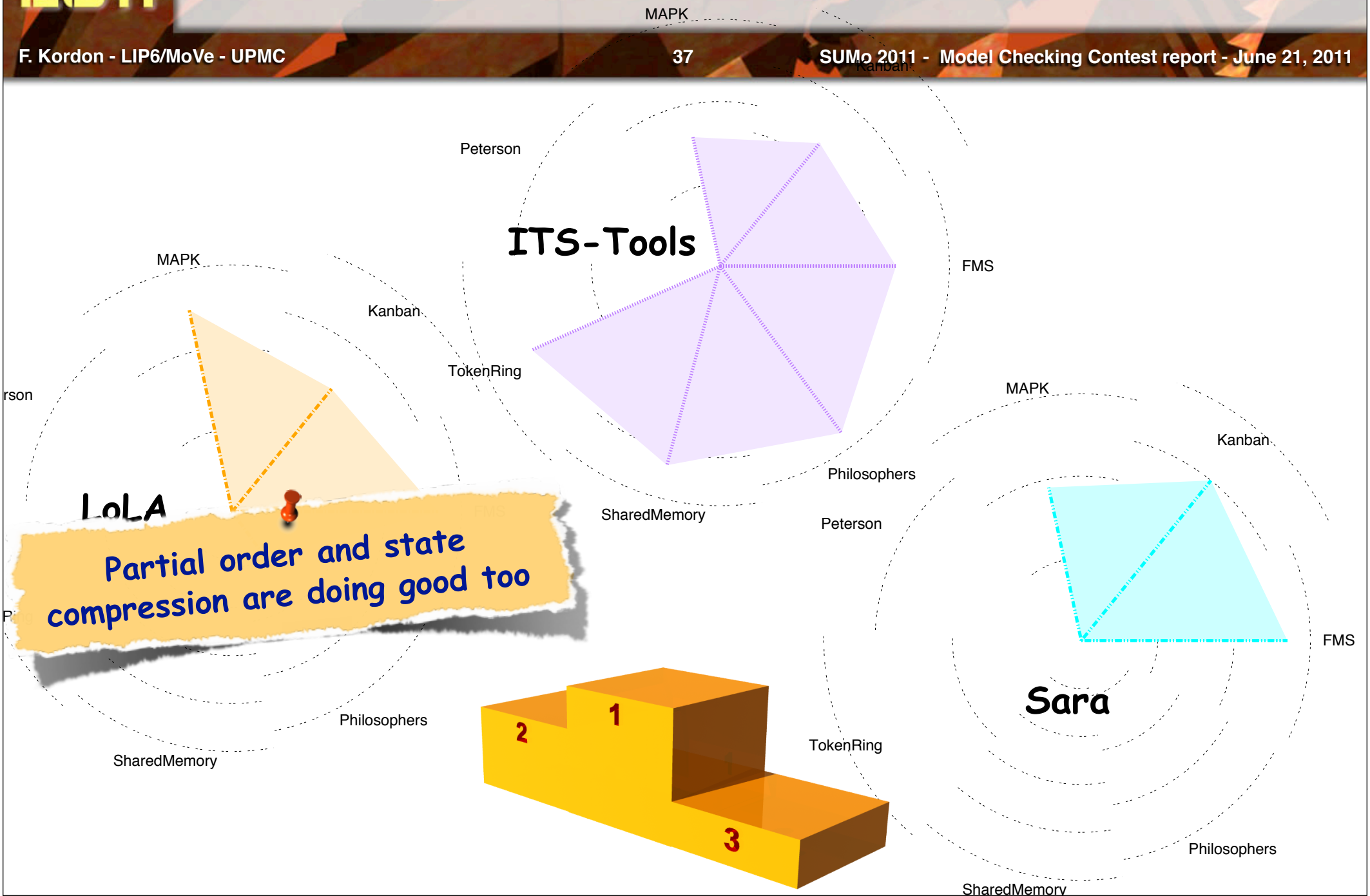
Reachability Analysis (Verified), Best Tools



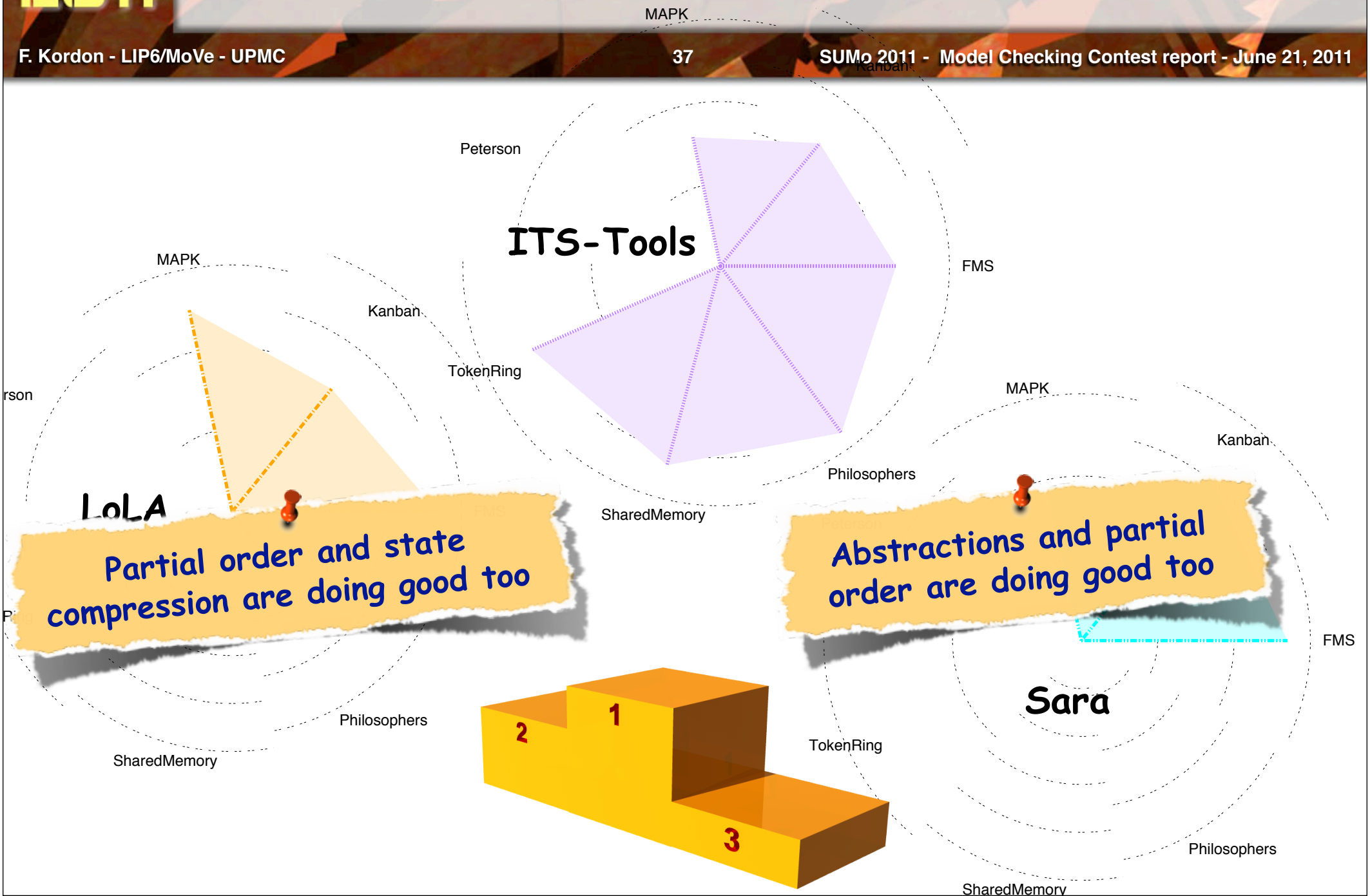
Reachability Analysis (Verified), Best Tools



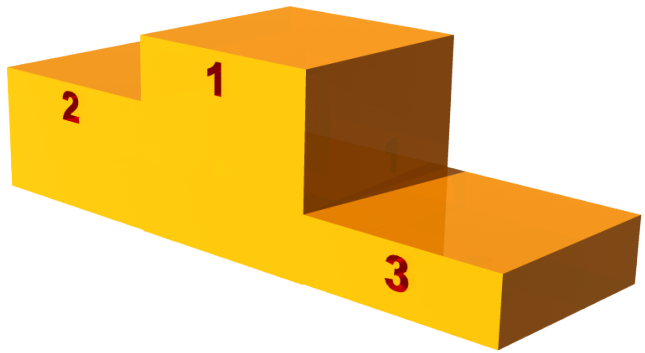
Reachability Analysis (Verified), Best Tools



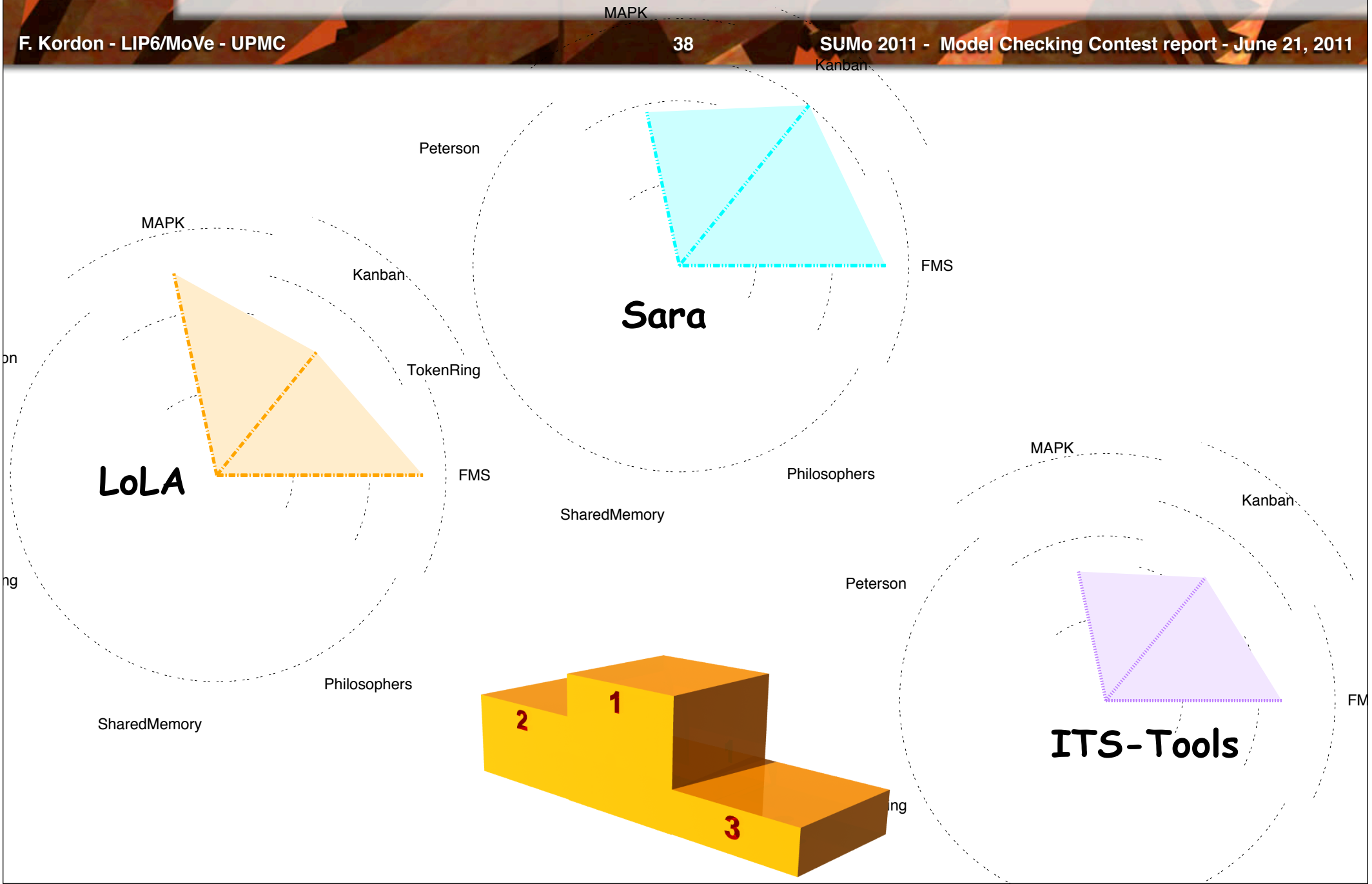
Reachability Analysis (Verified), Best Tools



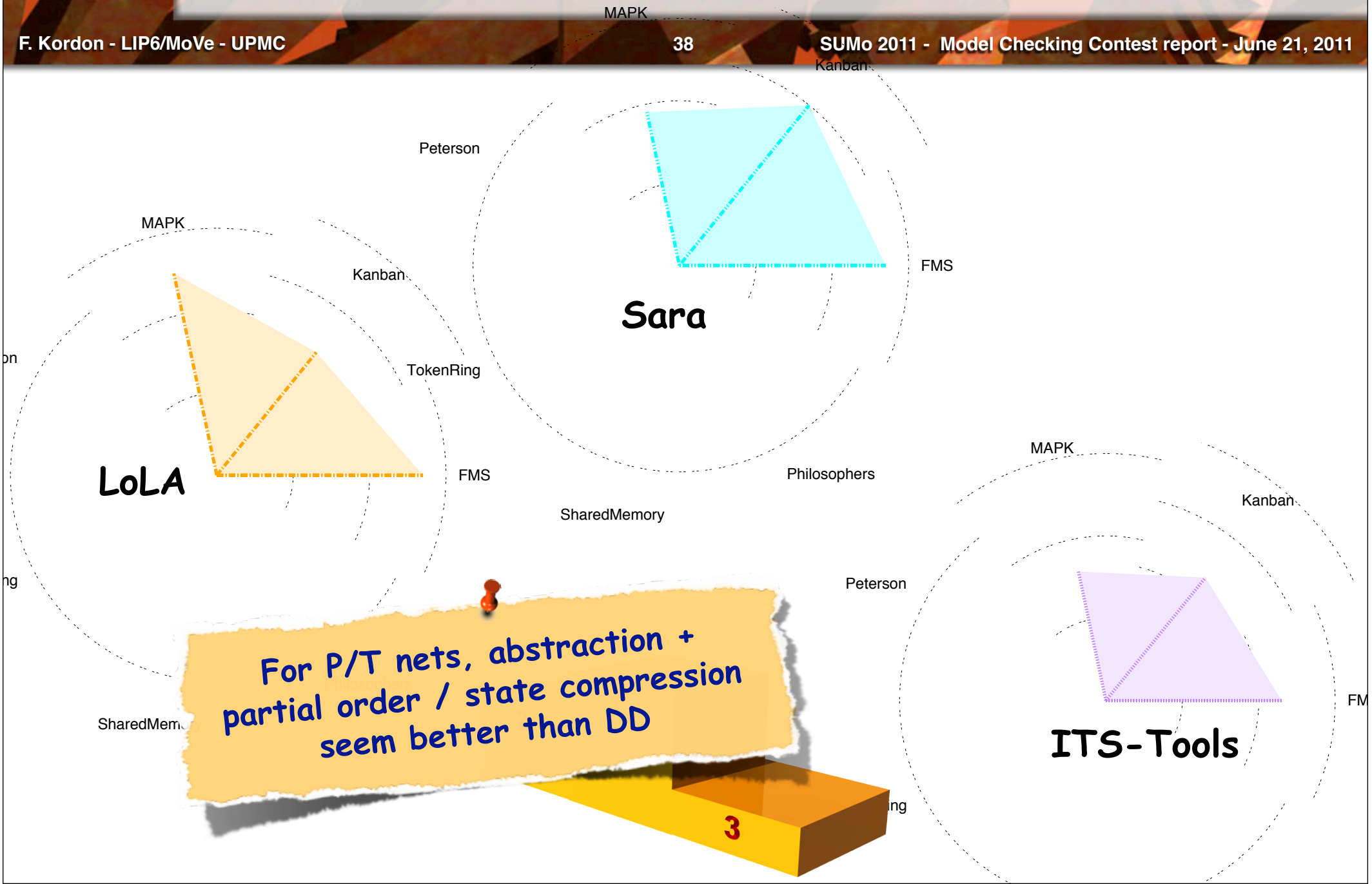
Reachability Analysis (Verified), Best Tools (P/T Nets)



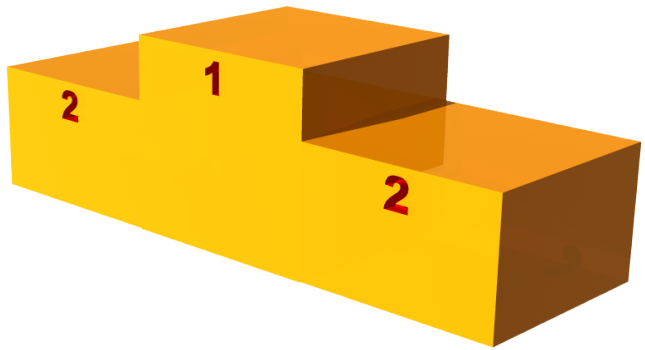
Reachability Analysis (Verified), Best Tools (P/T Nets)



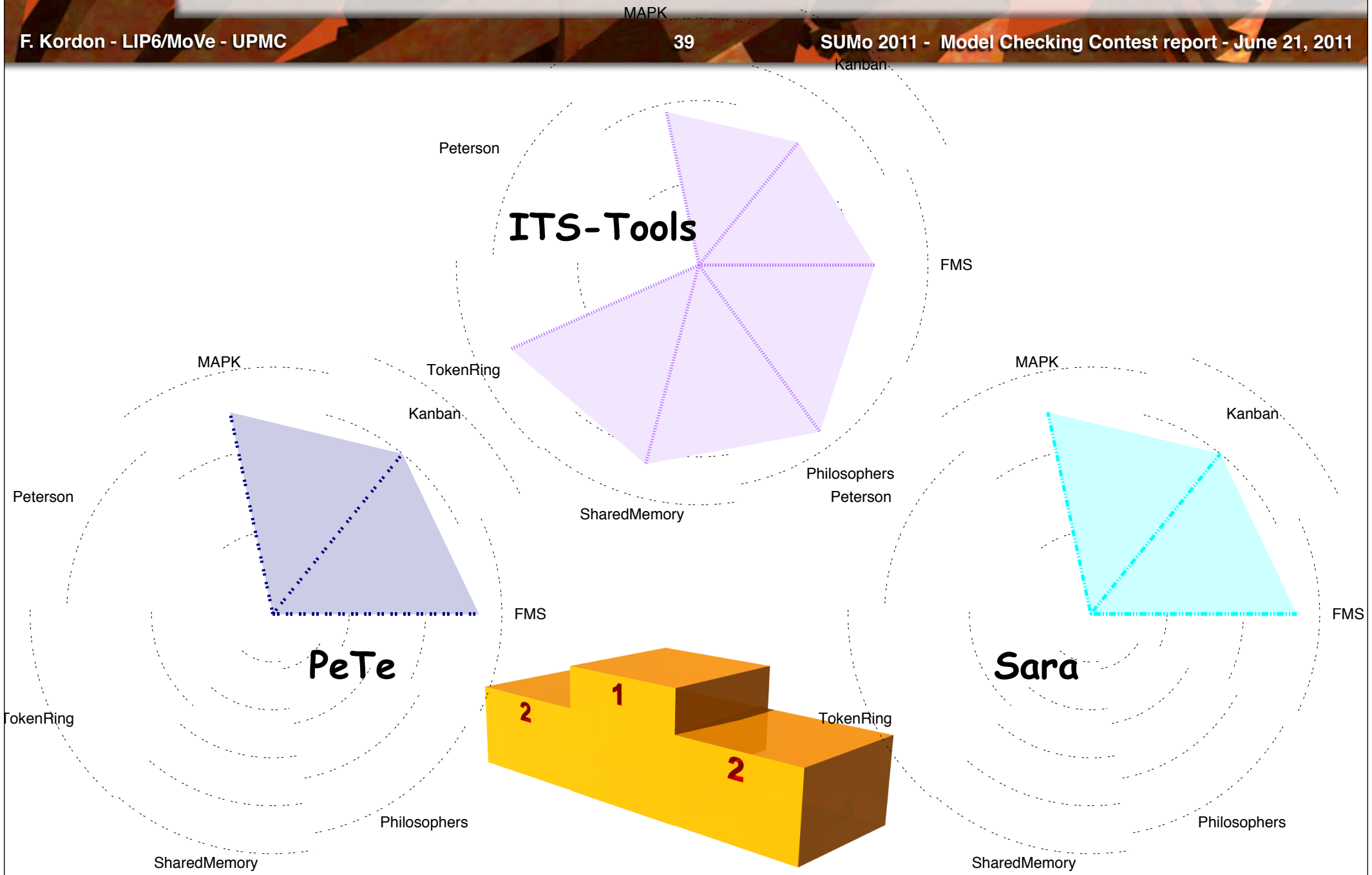
Reachability Analysis (Verified), Best Tools (P/T Nets)



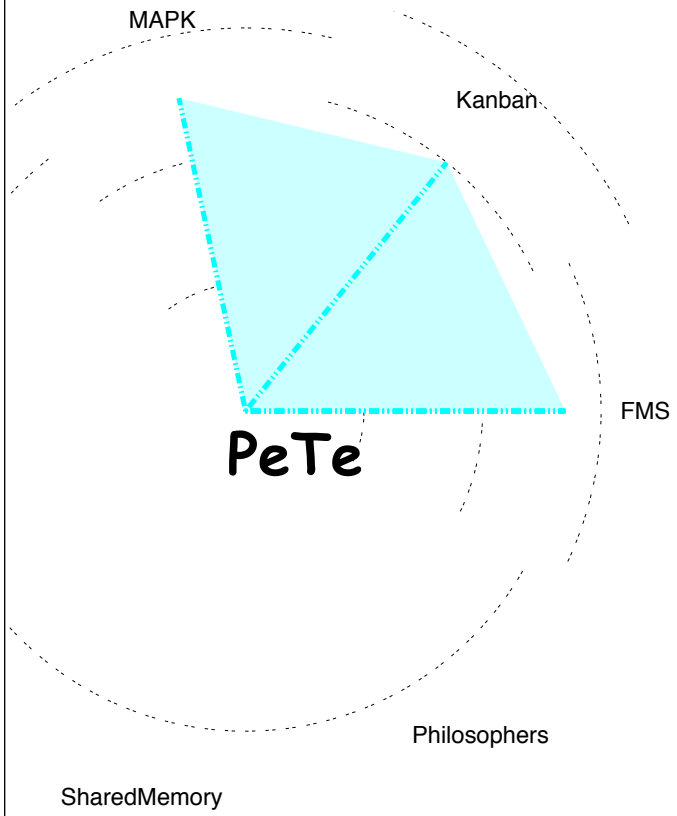
Reachability Analysis (Unverified), Best Tools

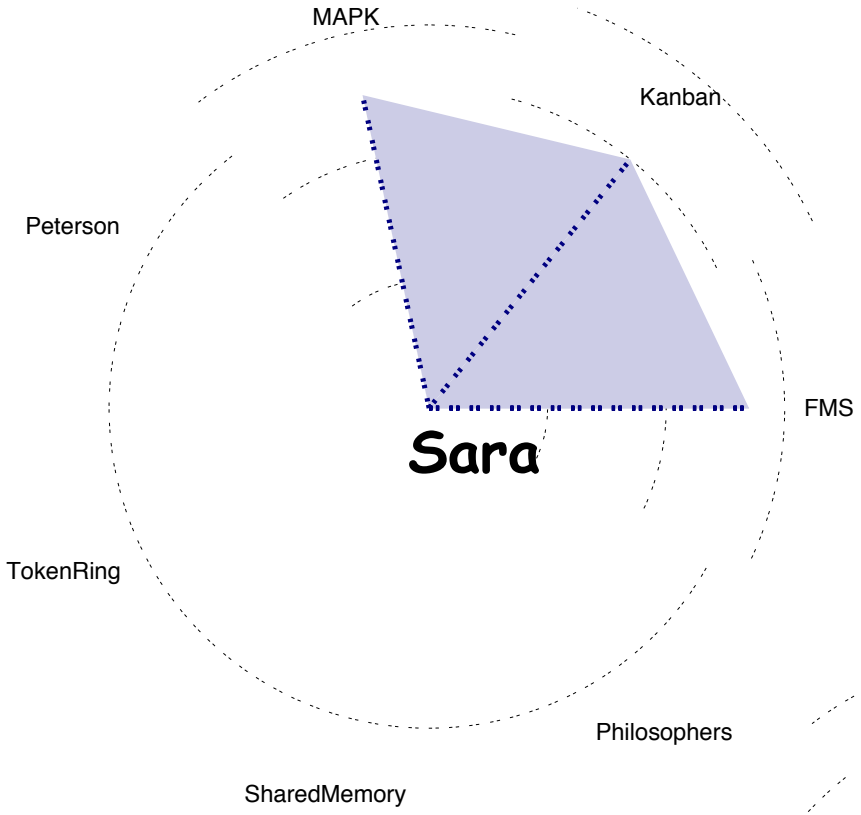


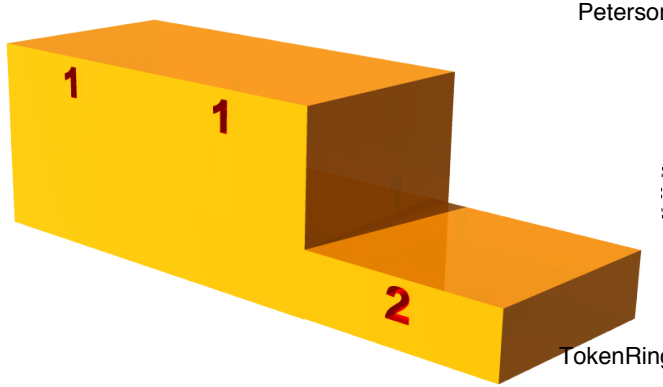
Reachability Analysis (Unverified), Best Tools

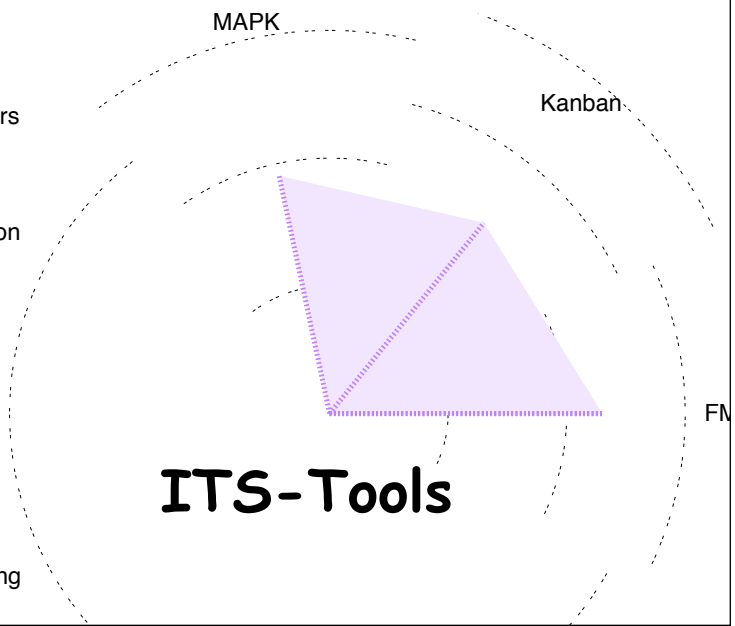


Reachability Analysis (Unverified), Best Tools (P/T Nets)

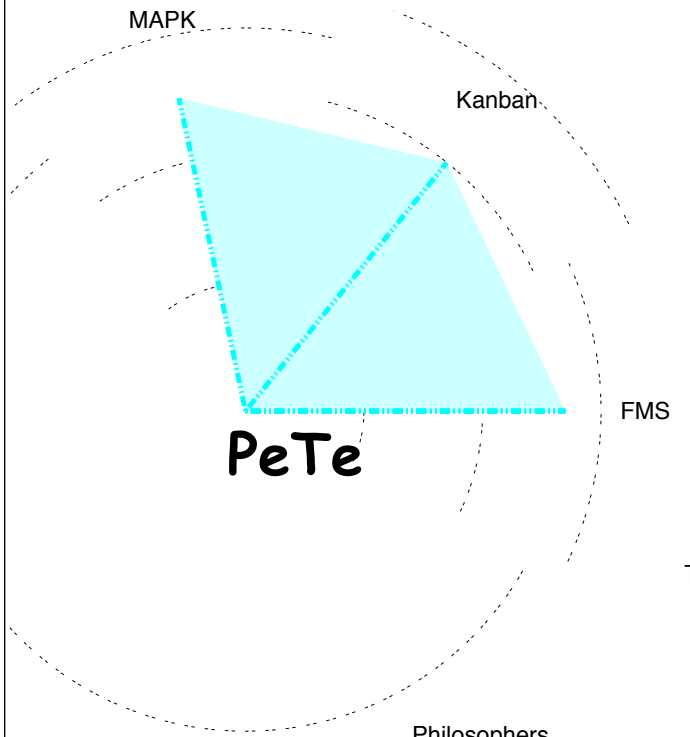


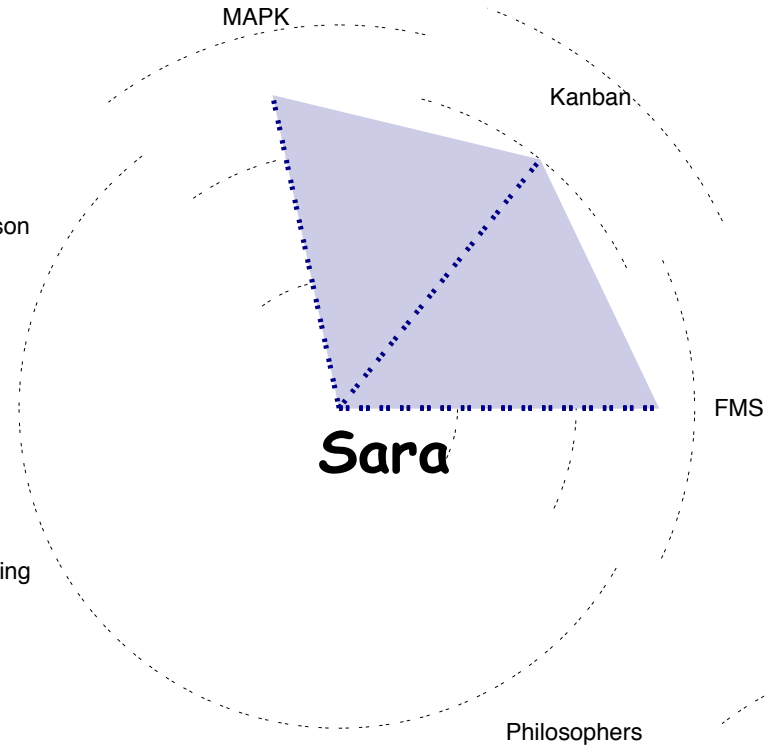


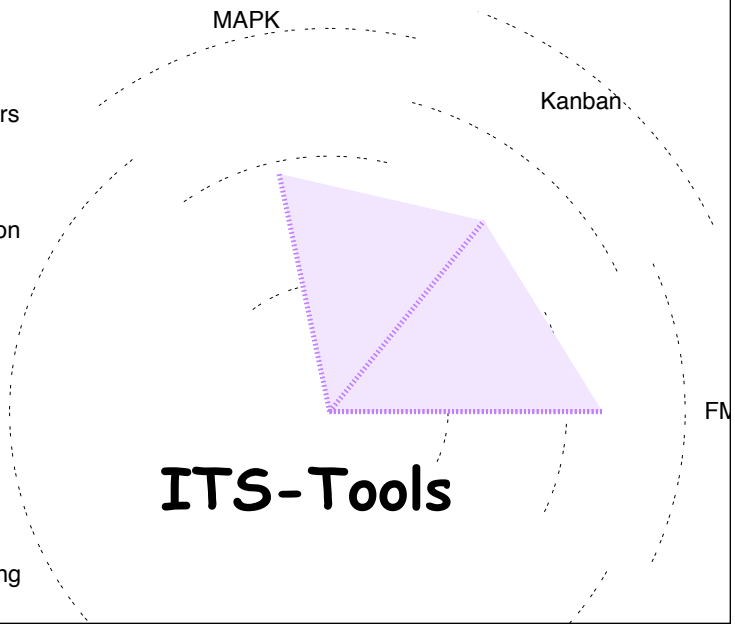




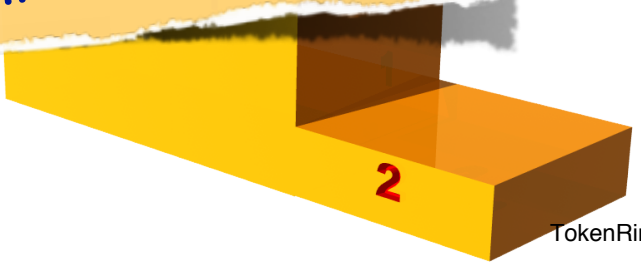
Reachability Analysis (Unverified), Best Tools (P/T Nets)





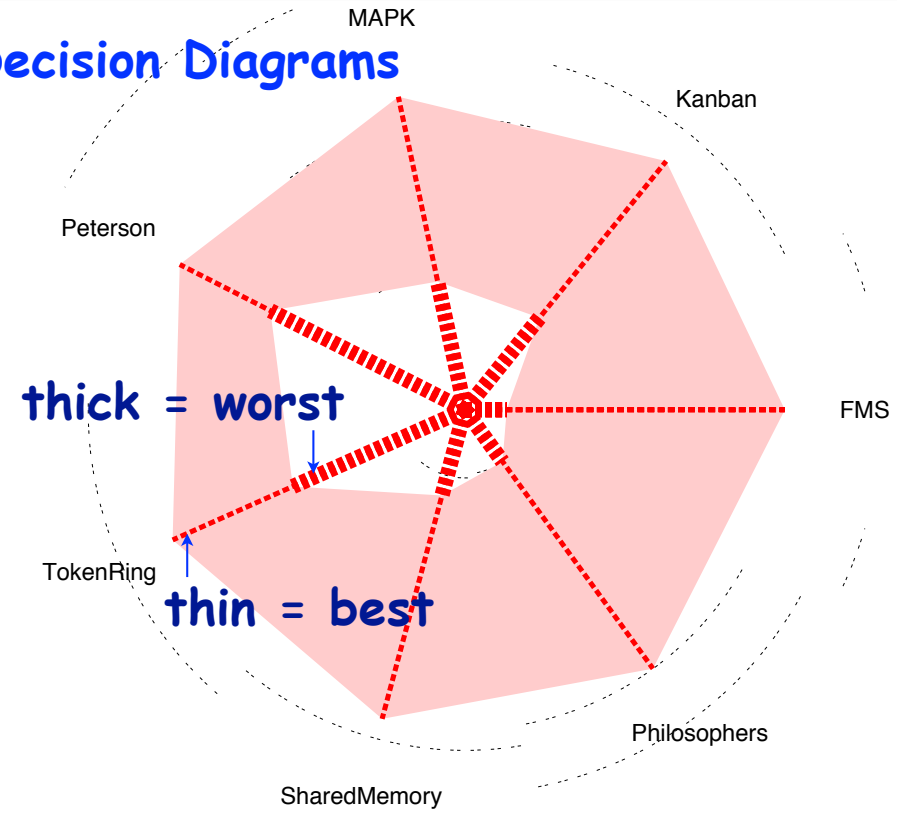


For P/T nets, abstraction + partial order/others seem better than DD

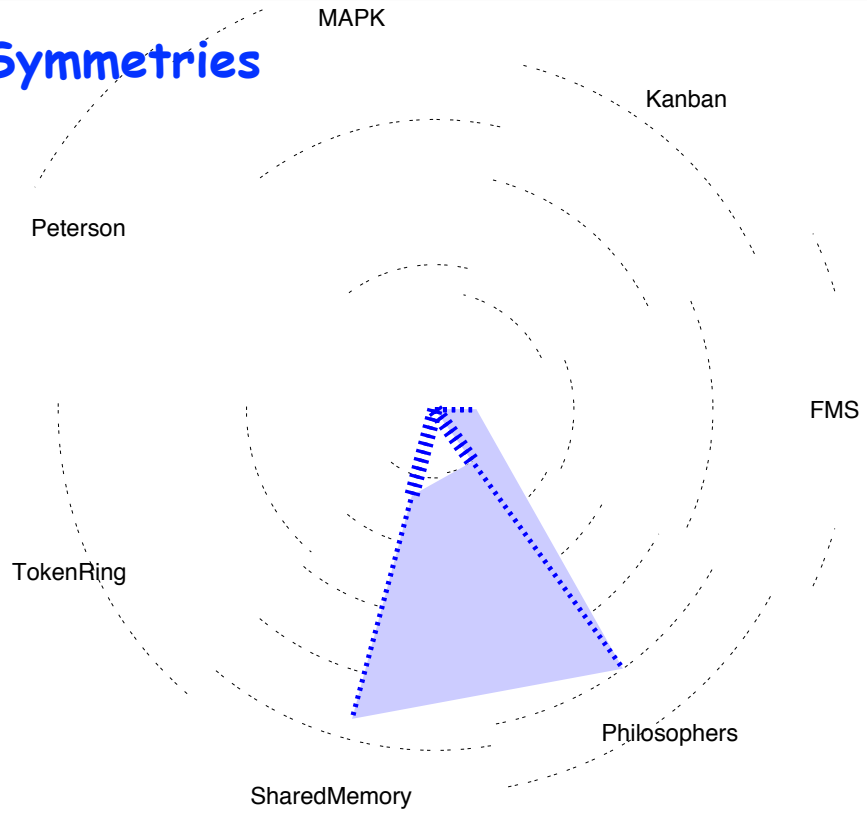


About Decision Diagrams and Symmetries

Decision Diagrams



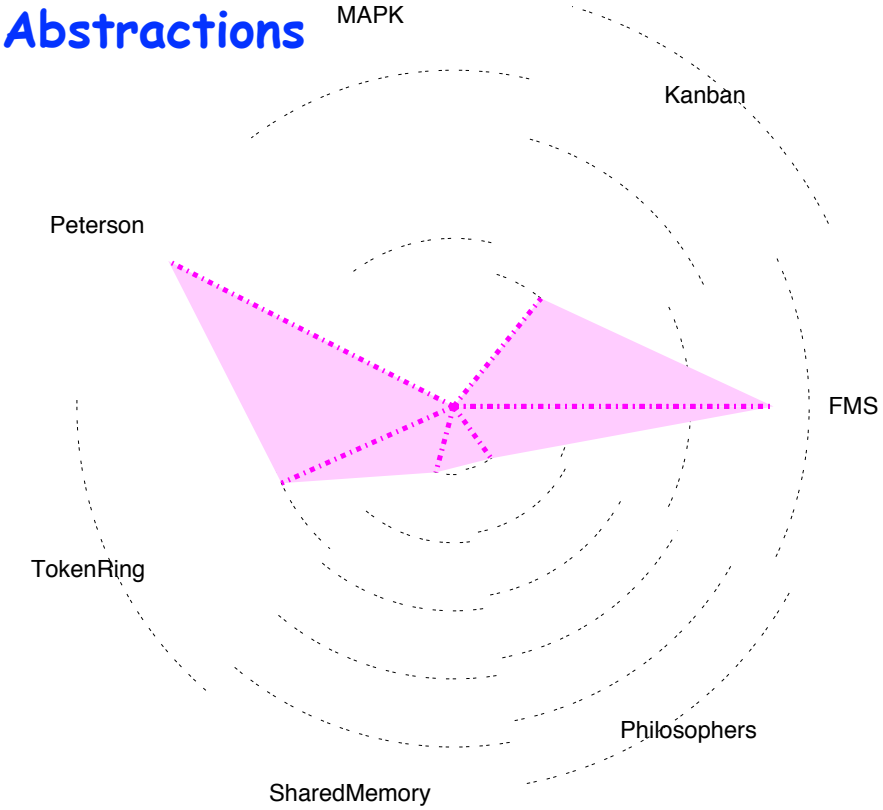
Symmetries



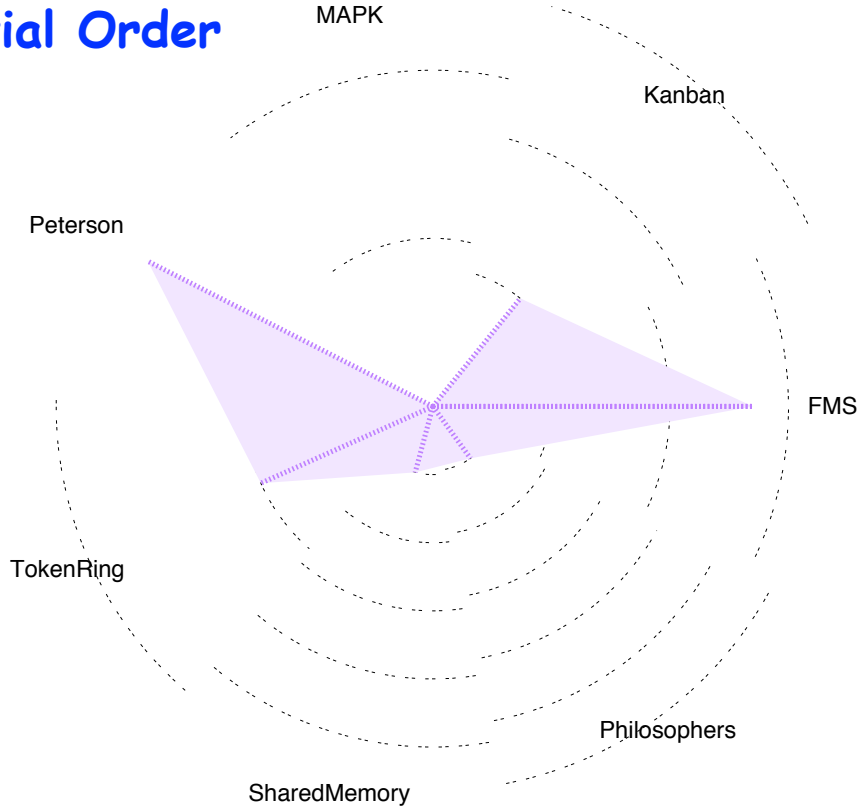
- «technique by technique» «radar»
  - Subtract maximum of worse tools with maximum of best tools
  - Shows progression for this technique
  - Here, for state space generation
- Stacking good for colored models
  - Especially regular ones
  - Good for: RG, DL, FOK, FNOK
- Makes less sense for P/T models
  - What about the model structure?

Abstractions and Partial Orders for Deadlock Detection

Abstractions



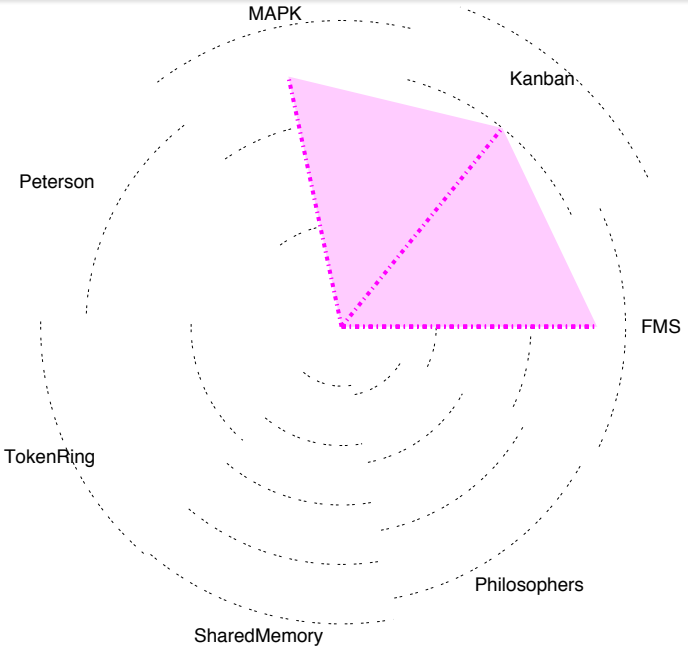
Partial Order



- Both techniques are efficient
  - No deadlock detection on MAPK
  - For Peterson, see remark to come
  - Seems to work on both P/T and Colored Models
- Good scalability when stacked
  - Sara are examples

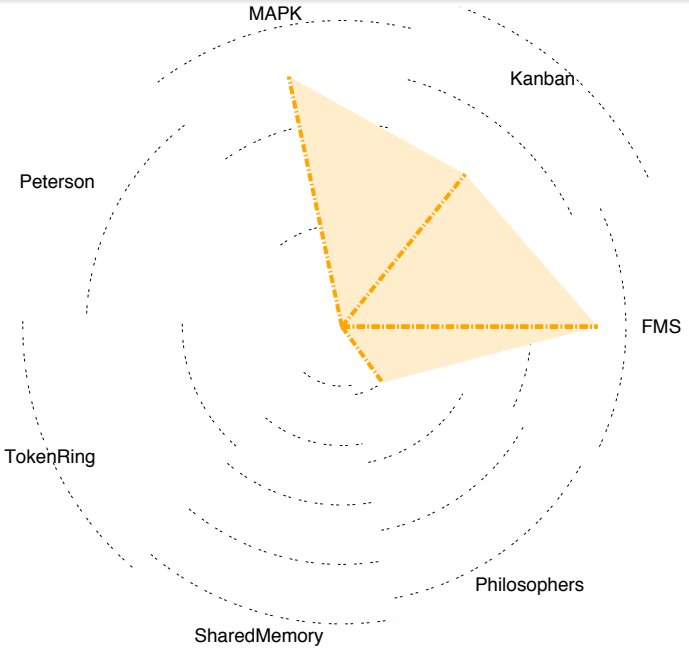
Formula evaluation...

- Abstraction work on both FOK and FNOK
  - But only experimented on P/T models this year



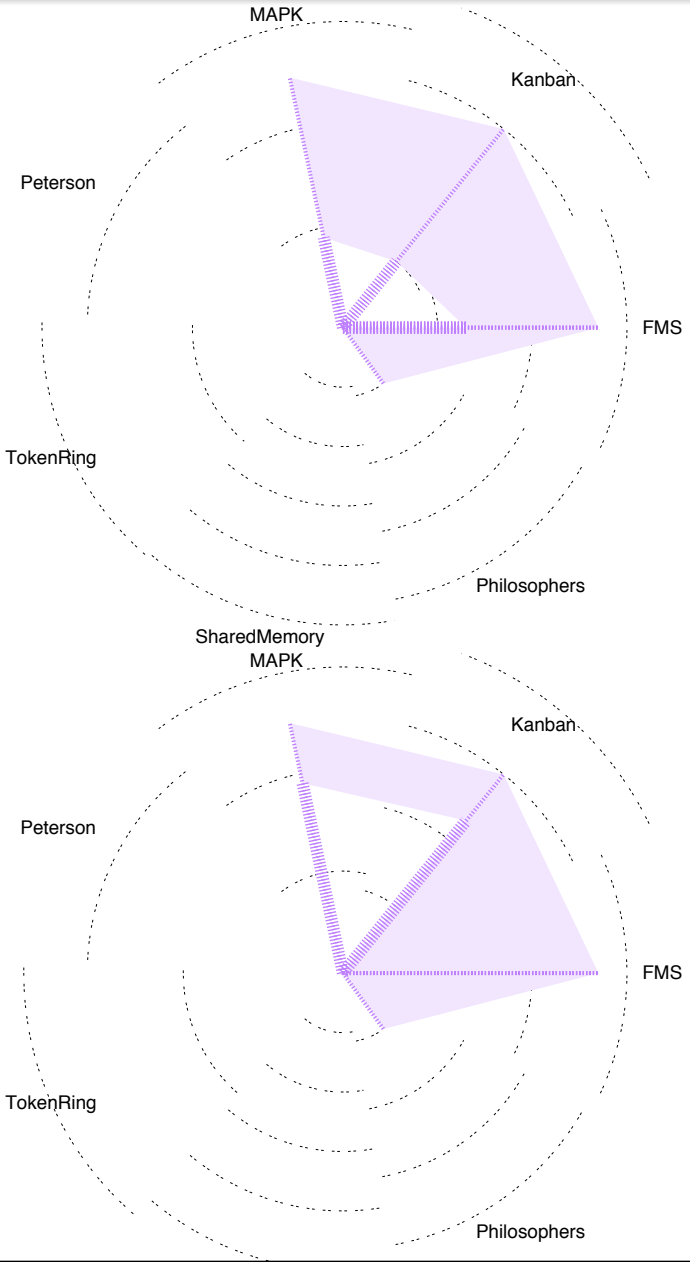
Formula evaluation...

- Abstraction work on both FOK and FNOK
  - But only experimented on P/T models this year
- State compression works well for FOK



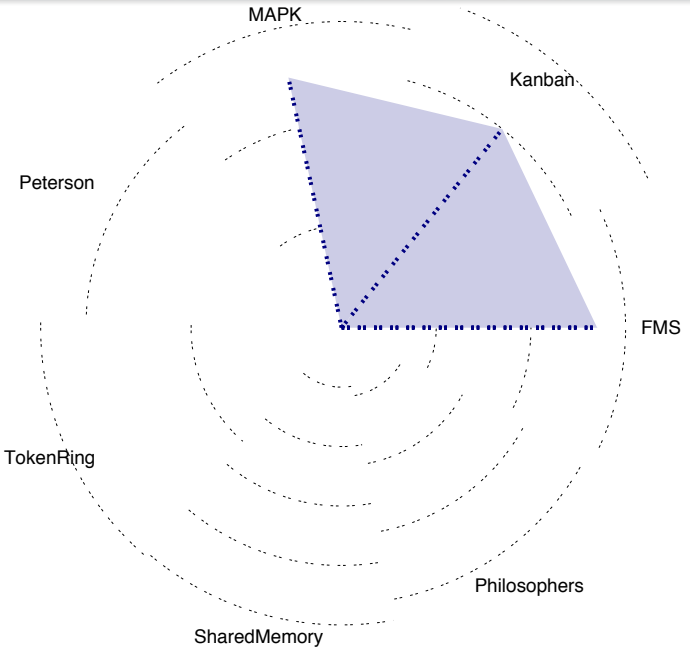
Formula evaluation...

- Abstraction work on both FOK and FNOK
  - But only experimented on P/T models this year
- State compression works well for FOK
- Partial orders work for formula evaluation
  - Slightly better for FNOK

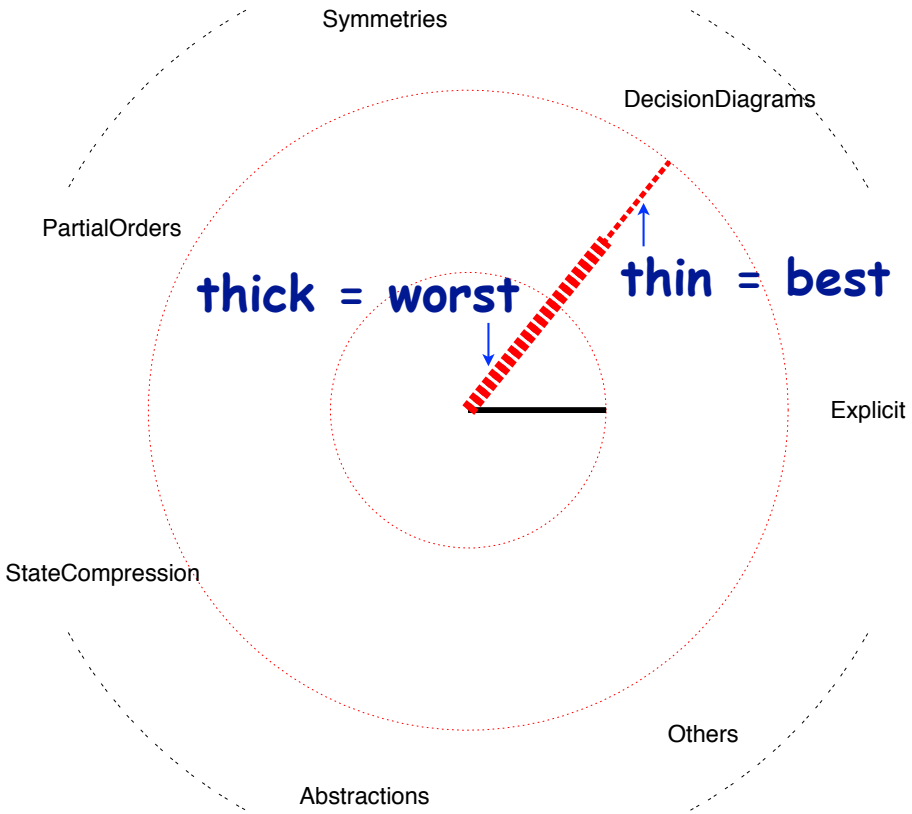


Formula evaluation...

- Abstraction work on both FOK and FNOK
  - But only experimented on P/T models this year
- State compression works well for FOK
- Partial orders work for formula evaluation
  - Slightly better for FNOK
- «Other Techniques» works too
  - Need to be refined
    - «other»?
  - Excellent for FNOK
    - structural evaluation of formula



Special Note on the Peterson Model



- The «technique by model» radar
  - Shows the appropriate technique for a given model
- The Peterson model seems very resistant
  - Techniques have difficulties to scale up

Very Last Remarks & questions
(from discussions with developers)

- Some tools are dedicated to specific classes
  - AlPiNA for Algebraic Nets
  - Crocodile for SN with Bags
- Some tool are more general
  - ACTIVITY-LOCAL supports multiple formalisms
- Memory measure to be refined
  - Strange behavior for PeTe?

These tools may not have been used in their best conditions...

Very Last Remarks & questions
(from discussions with developers)

- Some tools are dedicated to specific classes
  - AlPiNA for Algebraic Nets
  - Crocodile for SN with Bags
- Some tool are more general
  - ACTIVITY-LOCAL supports multiple formalisms
- Memory measure to be refined
  - Strange behavior for PeTe?

These tools may not have been used in their best conditions...

- Questions raised from this (partial) study
  - Could we stack reduction techniques with DD as for Symmetries?
  - Can abstractions be stacked with other techniques?
    - True for abstraction + DD (but not a competing tool)
  - Can partial order be stacked with other techniques?

Very Last Remarks & questions
(from discussions with developers)

- Some tools are dedicated to specific classes
  - AlPiNA for Algebraic Nets
  - Crocodile for SN with Bags
- Some tool are more general
  - ACTIVITY-LOCAL supports multiple formalisms
- Memory measure to be refined
  - Strange behavior for PeTe?

These tools may not have been used in their best conditions...

- Questions raised from this (partial) study
  - Could we stack reduction techniques with DD as for Symmetries?
  - Can abstractions be stacked with other techniques?
    - True for abstraction + DD (but not a competing tool)
  - Can partial order be stacked with other techniques?

- Lots of new stuff to experiment for the next MCC ;-)

MCC 2011

Concluding Remarks

What Could Come Out?

- A first attempt to evaluate model checkers with common criteria
- Some benchmarks (to be updated of course)
- A common language for queries? (to be updated of course)

- Some inputs for tool developers
  - Potential association of some techniques
  - Identification of some problems

- Toward adaptive model checking?
  - What technique is appropriate for a given type of model

  - Example: in the first modeling (debugging) phase, select a tool performant with unverified properties, then chose a tool performant with verified properties
    - Tool compatibilities issues then...

What Could Come Out?

- A first attempt to evaluate model checkers with common criteria
- Some benchmarks (to be updated of course)
- A common language for queries? (to be updated of course)

- Some inputs for tool developers
  - Potential association of some techniques
  - Identification of some problems

Profile tools from «neutral» benchmark (as for LTL to Büchi automata)

- Toward adaptive model checking?
  - What technique is appropriate for a given type of model
  - Example: in the first modeling (debugging) phase, select a tool performant with unverified properties, then chose a tool performant with verified properties
    - Tool compatibilities issues then...

Next Edition in 2012?

- As a standalone event?
  - Summary + presentation of some interesting techniques and approaches
- New stuff to be proposed in the next editions
  - More models: safe P/T, unsafe Colored, «industrial»
  - More Petri Net classes: timed PN? stochastic PN?
  - More properties: LTL? CTL?
  - Better identification of techniques
  - Enhanced confinement and measure environment
  - Do we suggest PNML as the only input for tools?

Downloading Participating Tools

- ACTIVITY-LOCAL
  - http://dblp.uni-trier.de/rec/bibtex/conf/mmb/LampkaS06 (reference but not the tool itself )
- AlPiNA
  - http://alpina.unige.ch
- Crocodile
  - http://move.lip6.fr/software/Crocodile
- ITS-Tools
  - http://ddd.lip6.fr
- LoLA
  - http://www.informatik.uni-rostock.de/tpp/lola/
- PNXDD
  - https://srcdev.lip6.fr/trac/research/NEOPPOD/wiki/pnxdd
- PeTe
  - https://github.com/jopsen/PeTe
- Sara
  - http://www.service-technology.org/tools/download
- YASPA
  - http://www.tik.ee.ethz.ch/~klampka
- helena
  - http://helena-mc.sourceforge.net