Feature-based Testing of SPLs: Pairwise and Beyond

Gilles Perrouin

(and many others :))

MOTIVATION

If there is a better reason to paddle, I don't know what it is.

Context

Context

- ★ The SPL paradigm promises high quality software through systematic assets reuse

Context

- ★ The SPL paradigm promises high quality software through systematic assets reuse

Context

★ The SPL paradigm promises high quality software through systematic assets reuse

★ Q: How to ensure such quality ?

Context

★ The SPL paradigm promises high quality software through systematic assets reuse

★ Q: How to ensure such quality ?

Context

★ The SPL paradigm promises high quality software through systematic assets reuse

★ Q: How to ensure such quality ?

★ Need for SPL QA

- Model checking [Asirelli2011, Classen2010, Classen2011, Fischbein2006, Gruler2008, Lauenroth2009, Li2002]
- Testing [Oster2010, Perrouin2011, Weiβleder2010]

Features: Render unto Caesar...

VaMoS

Fourth International Workshop on
Variability Modelling of Software-intensive Systems
"Celebrating 20 Years of Feature Models"

Johannes Kepler University Linz, Austria — January 27-29, 2010

~~20~~22 years of features...

~~20~~ 22 years of features...

- ★ As many meanings as there are SPL researchers [Classen2008] :)

~~20~~ 22 years of features...

- ★ As many meanings as there are SPL researchers [Classen2008] :)
- ★ Distinguishable units of interest => abstract
  - ‣ Mapping features to models if needed

~~20~~ 22 years of features...

- ★ As many meanings as there are SPL researchers [Classen2008] :)
- ★ Distinguishable units of interest => abstract
  - ‣ Mapping features to models if needed
- ★ Organised in Feature Models (FM)
  - ‣ Graphical Notation: FODA-style [Kang1990]
  - ‣ Textual: Guidsl [Batory2005], TVL [Classen2010b]...

~~20~~ 22 years of features...

- ★ As many meanings as there are SPL researchers [Classen2008] :)
- ★ Distinguishable units of interest => abstract
  - Mapping features to models if needed
- ★ Organised in Feature Models (FM)
  - Graphical Notation: FODA-style [Kang1990]
  - Textual: Guidsl [Batory2005], TVL [Classen2010b]...
- ★ Formalisations exist [Batory2005,Czarnecki2007, Schobbens2007]

~~20~~ 22 years of features...

- ★ As many meanings as there are SPL researchers [Classen2008] :)
- ★ Distinguishable units of interest => abstract
  - Mapping features to models if needed
- ★ Organised in Feature Models (FM)
  - Graphical Notation: FODA-style [Kang1990]
  - Textual: Guidsl [Batory2005], TVL [Classen2010b]...
- ★ Formalisations exist [Batory2005,Czarnecki2007, Schobbens2007]
- ★ Automated analyses [Benavides2010]

~~20~~ 22 years of features...

- ★ As many meanings as there are SPL researchers [Classen2008] :)
- ★ Distinguishable units of interest => abstract
  - Mapping features to models if needed
- ★ Organised in Feature Models (FM)
  - Graphical Notation: FODA-style [Kang1990]
  - Textual: Guidsl [Batory2005], TVL [Classen2010b]...
- ★ Formalisations exist [Batory2005,Czarnecki2007, Schobbens2007]
- ★ Automated analyses [Benavides2010]
- ★ Enables Model-Driven QA of SPL :)

Once upon a time... (2009)

Once upon a time... (2009)

- ★ There was a postdoc working on (structural) product derivation for SPLs...
  - Compositional approach [Perrouin2008]

Once upon a time... (2009)

- ★ There was a postdoc working on (structural) product derivation for SPLs...
  - Compositional approach [Perrouin2008]
- ★ Q: How to design model fragments so that they compose well together ?
  - Methodological hints are insufficient
  - Need for an automated approach to validate SPL models...

Once upon a time... (2009)

- ★ There was a postdoc working on (structural) product derivation for SPLs...
  - Compositional approach [Perrouin2008]
- ★ Q: How to design model fragments so that they compose well together ?
  - Methodological hints are insufficient
  - Need for an automated approach to validate SPL models...

Once upon a time... (2009)

- ★ There was a postdoc working on (structural) product derivation for SPLs...
  - Compositional approach [Perrouin2008]
- ★ Q: How to design model fragments so that they compose well together ?
  - Methodological hints are insufficient
  - Need for an automated approach to validate SPL models...
- ★ Testing view: Extract relevant configurations of the SPL and build them (composition = oracle)

Initial Requirements

Initial Requirements

- ★ Model-based Testing (FM = Model) [Utting2006]
  - Integration in the whole SPL lifecycle
  - Automation through model transformations

Initial Requirements

- ★ Model-based Testing (FM = Model) [Utting2006]
  - Integration in the whole SPL lifecycle
  - Automation through model transformations

Initial Requirements

- ★ Model-based Testing (FM = Model) [Utting2006]
  - Integration in the whole SPL lifecycle
  - Automation through model transformations
- ★ Product-by-Product: 2^N tests for N features :)

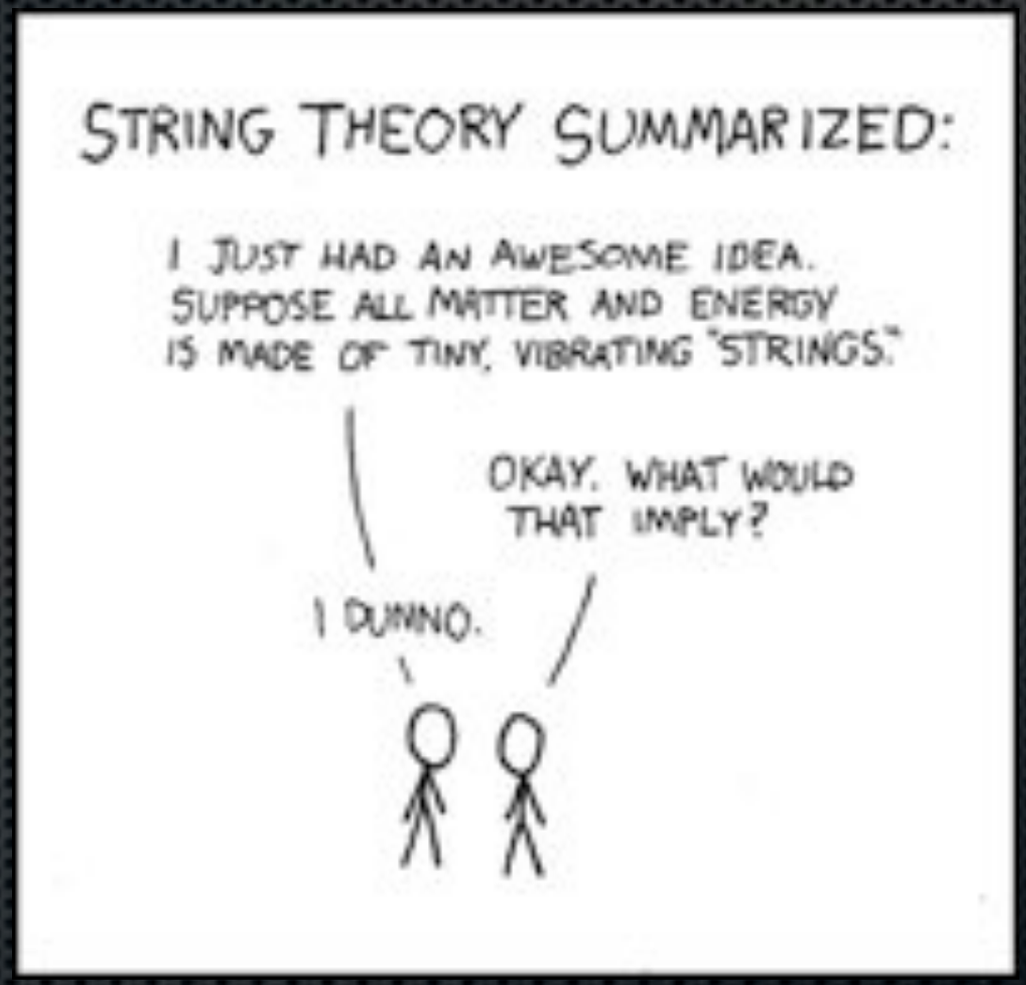
Initial Requirements

- ★ Model-based Testing (FM = Model) [Utting2006]
  - Integration in the whole SPL lifecycle
  - Automation through model transformations
- ★ Product-by-Product: 2^N tests for N features :)

Initial Requirements

- ★ Model-based Testing (FM = Model) [Utting2006]
  - Integration in the whole SPL lifecycle
  - Automation through model transformations
- ★ Product-by-Product: 2^N tests for N features :)
- ★ Assumed no a priori knowledge of the SPL
  - Incremental testing [Uzu08] infeasible (where to start ?)
  - Needed to cope with combinatorial explosion ***and*** interactions

Initial Vision



Combinatorial Interaction Testing

Combinatorial Interaction Testing

- ★ Most bugs are provoked by a small number of interactions [Kuhn2004]
- ★ T-wise coverage criteria: "all interactions of size t must be covered in test cases at least once"
- ★ T= [1..6]. T= 2 (pairwise) often enough (>70%)
- ★ Pros:
  - Addresses the *feature interaction problem*
  - Small test suites (compared to 10^X possible tests)
- ★ Cons (for SPL)
  - Poor support for constraints
  - Limited SPL support (2009) [Cohen2006,Cohen2007]

T-wise SPL Testing as a SAT problem

T-wise SPL Testing as a SAT problem

- ★ MBT approach considering FMs as inputs
  - ‣ Viewed as a set of constraints between boolean features

T-wise SPL Testing as a SAT problem

- ★ MBT approach considering FMs as inputs
  - ‣ Viewed as a set of constraints between boolean features
- ★ T-wise
  - ‣ Can be seen as a SAT problem: “Set of valid products that satisfy the conjunction of all t-tuples of features”

T-wise SPL Testing as a SAT problem

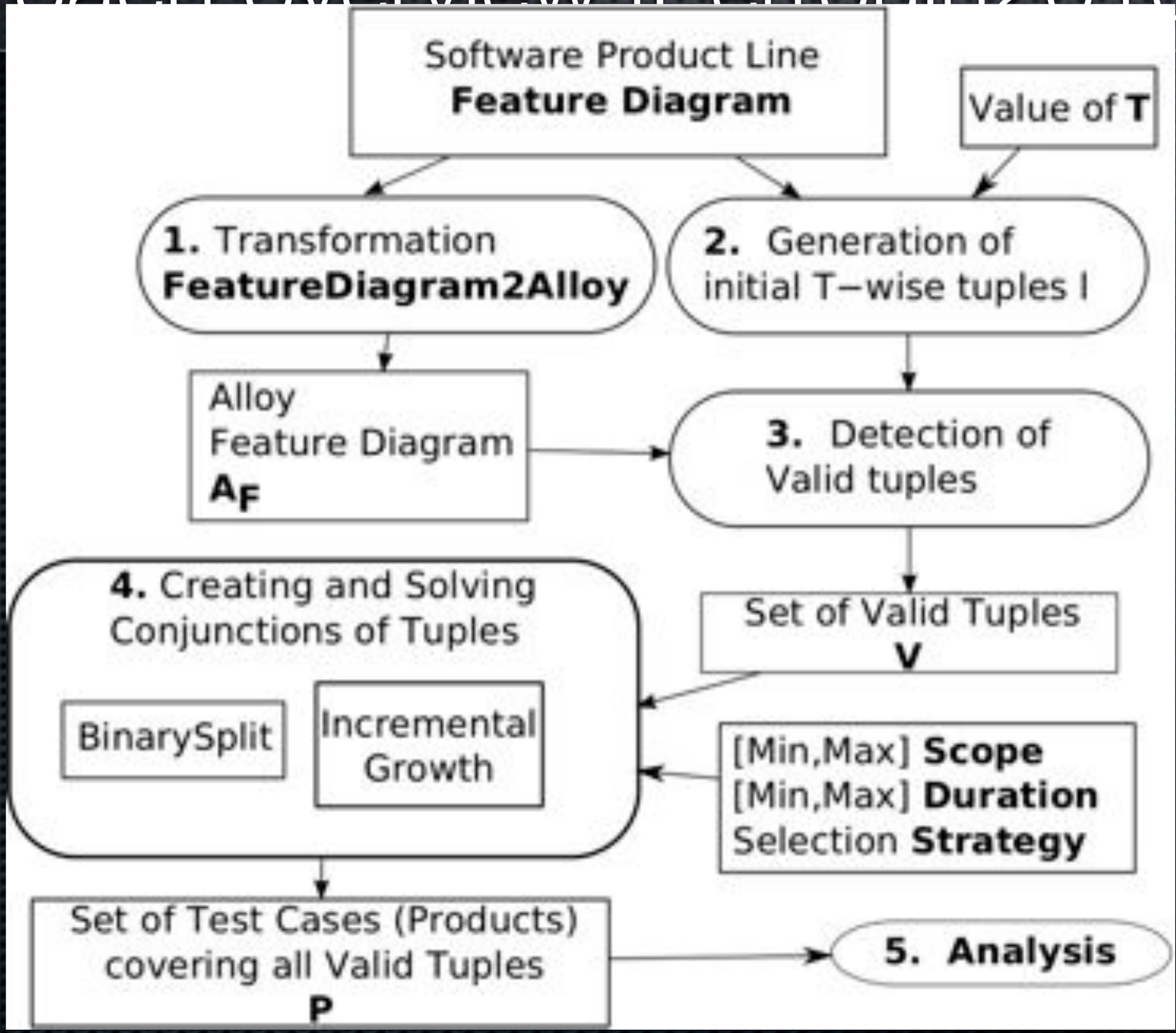
- ★ MBT approach considering FMs as inputs
  - ‣ Viewed as a set of constraints between boolean features
- ★ T-wise
  - ‣ Can be seen as a SAT problem: “Set of valid products that satisfy the conjunction of all t-tuples of features”
- ★ Rely on SAT solvers for SPL T-wise testing

T-wise SPL Testing as a SAT problem

- ★ MBT approach considering FMs as inputs
  - Viewed as a set of constraints between boolean features
- ★ T-wise
  - Can be seen as a SAT problem: “Set of valid products that satisfy the conjunction of all t-tuples of features”
- ★ Rely on SAT solvers for SPL T-wise testing
- ★ Questions
  - How to encode the FM + T-wise problems from higher models ?
  - Scalability (NP-Complete problem) ?

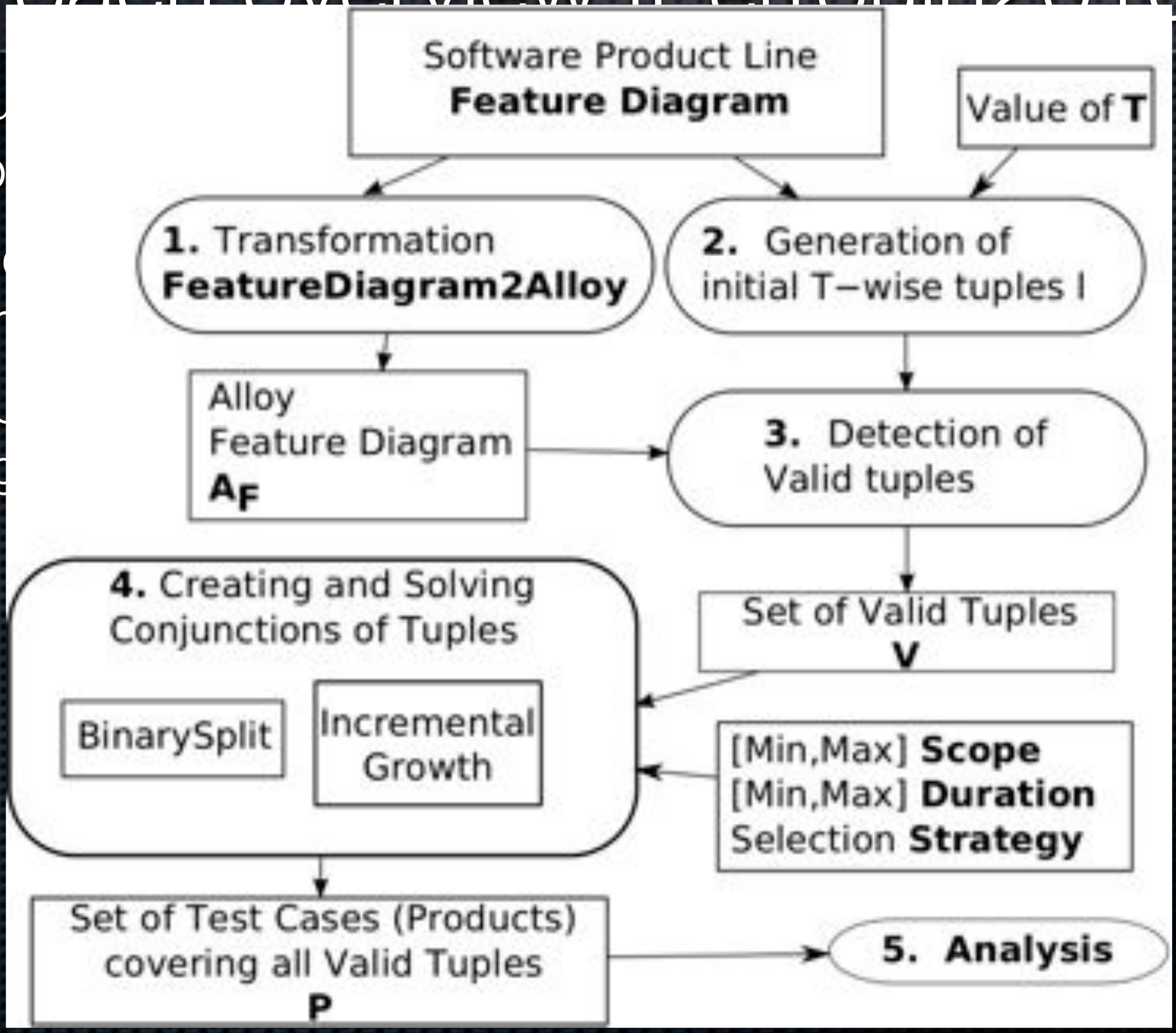
Approach overview [Perrouin2010]

Approach overview [Perrouin2010]



Approach overview [Perrouin2010]

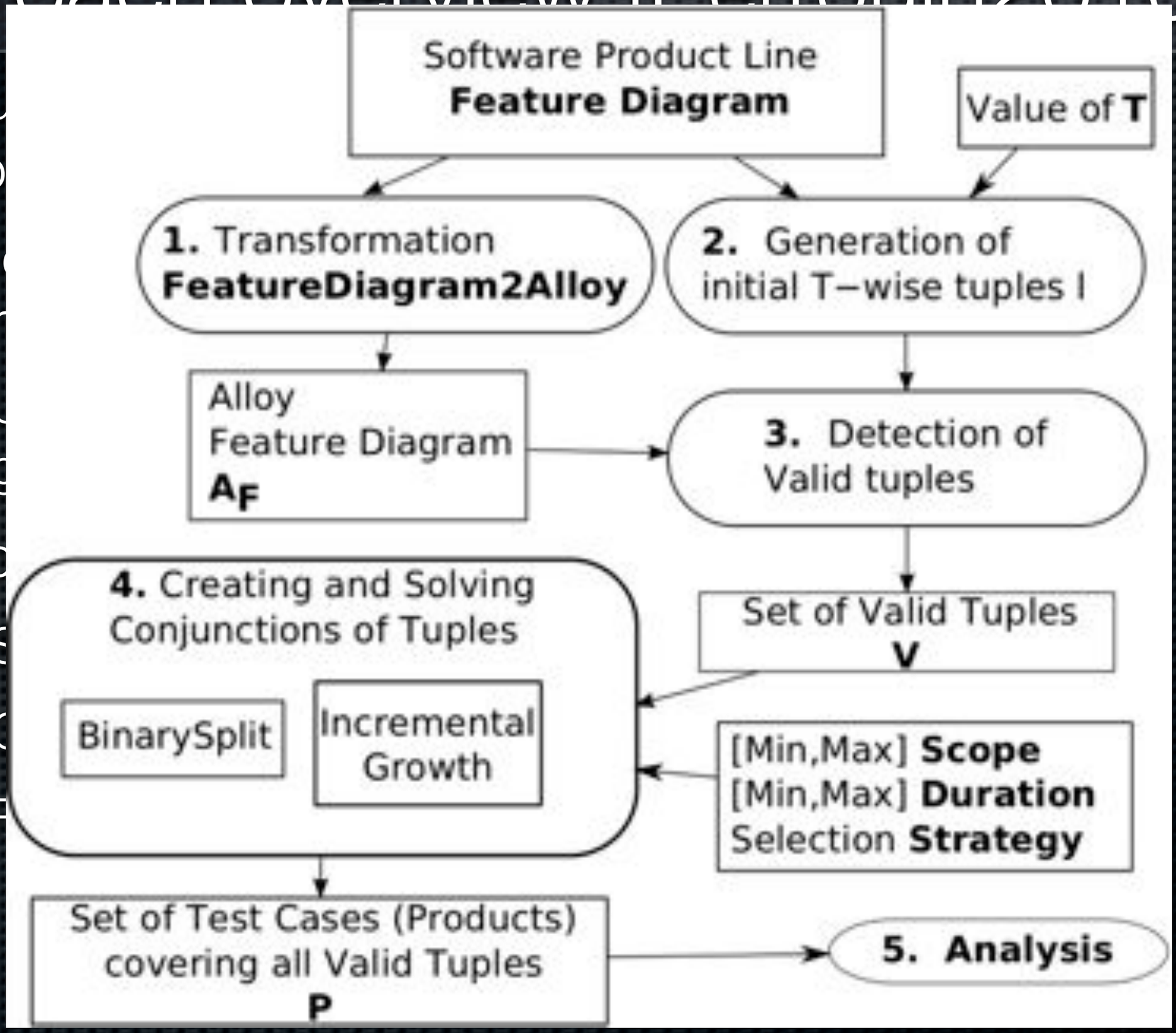
- ★ Au
  so
  - I
    r
  - U
    g



Approach overview [Perrouin2010]

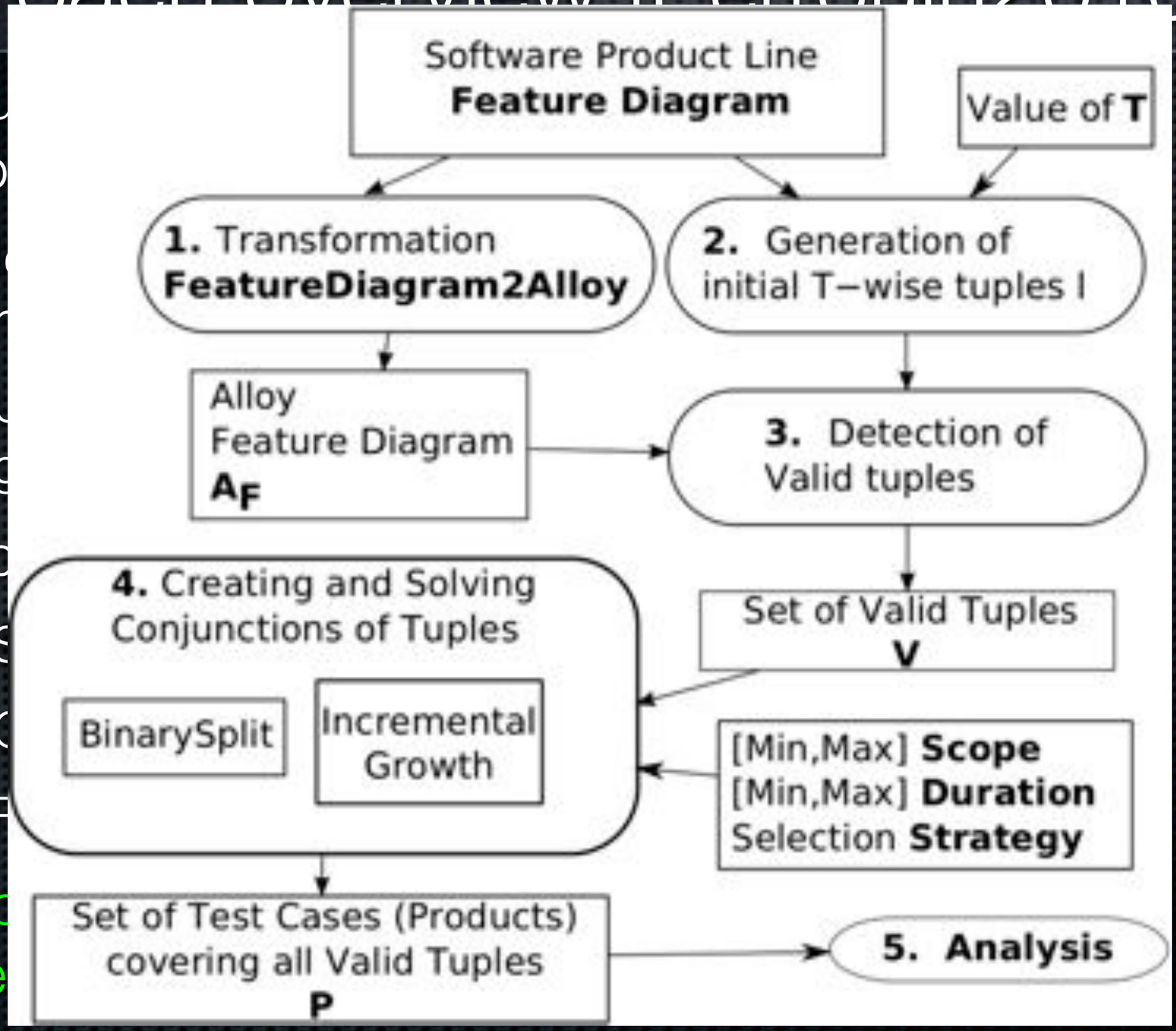
★ Au
so
- I
r
- U
g

★ So
- s
- C
- P



Approach overview [Perrouin2010]

- ★ Au
  so
  - I
    n
  - U
    g
- ★ So
  - s
  - C
  - P
- ★ Co
  se



Experimentations

Evaluating T-wise Generation

$$Sim(tc_i, tc_j) = \frac{\|Tci_v \cap Tcj_v\|}{\|Tci_v \cup Tcj_v\|}$$

Evaluating T-wise Generation

★ Generation time

$$Sim(tc_i, tc_j) = \frac{\|Tci_v \cap Tcj_v\|}{\|Tci_v \cup Tcj_v\|}$$

Evaluating T-wise Generation

- ★ Generation time
- ★ Generated test suite size

$$Sim(tc_i, tc_j) = \frac{\|Tci_v \cap Tcj_v\|}{\|Tci_v \cup Tcj_v\|}$$

Evaluating T-wise Generation

- ★ Generation time
- ★ Generated test suite size
- ★ T-tuple occurrence: *how many times a given T-tuple appears ?*

$$Sim(tc_i, tc_j) = \frac{\|Tci_v \cap Tcj_v\|}{\|Tci_v \cup Tcj_v\|}$$

Evaluating T-wise Generation

- ★ Generation time
- ★ Generated test suite size
- ★ T-tuple occurrence: *how many times a given T-tuple appears ?*
- ★ Number of duplicates *(engendered by "divide and compose" strategies)*

$$Sim(tc_i, tc_j) = \frac{\|Tci_v \cap Tcj_v\|}{\|Tci_v \cup Tcj_v\|}$$

Evaluating T-wise Generation

- ★ Generation time
- ★ Generated test suite size
- ★ T-tuple occurrence: *how many times a given T-tuple appears ?*
- ★ Number of duplicates *(engendered by "divide and compose" strategies)*
- ★ Similarity: *how different are my generated test cases ?*

$$Sim(tc_i, tc_j) = \frac{\|Tci_v \cap Tcj_v\|}{\|Tci_v \cup Tcj_v\|}$$

Evaluating T-wise Generation

- ★ Generation time
- ★ Generated test suite size
- ★ T-tuple occurrence: *how many times a given T-tuple appears ?*
- ★ Number of duplicates *(engendered by "divide and compose" strategies)*
- ★ Similarity: *how different are my generated test cases ?*

$$Sim(tc_i, tc_j) = \frac{\|Tci_v \cap Tcj_v\|}{\|Tci_v \cup Tcj_v\|}$$

Evaluating T-wise Generation

- ★ Generation time
- ★ Generated test suite size
- ★ T-tuple occurrence: *how many times a given T-tuple appears ?*
- ★ Number of duplicates *(engendered by "divide and compose" strategies)*
- ★ Similarity: *how different are my generated test cases ?*

$$Sim(tc_i, tc_j) = \frac{\|Tci_v \cap Tcj_v\|}{\|Tci_v \cup Tcj_v\|}$$

Evaluating T-wise Generation

- ★ Generation time
- ★ Generated test suite size
- ★ T-tuple occurrence: *how many times a given T-tuple appears ?*
- ★ Number of duplicates *(engendered by "divide and compose" strategies)*
- ★ Similarity: *how different are my generated test cases ?*

$$Sim(tc_i, tc_j) = \frac{\|Tci_v \cap Tcj_v\|}{\|Tci_v \cup Tcj_v\|}$$

- ★ Tci_v : *Variant features [Benavides2010] of test case 'i'*

Comparing Pairwise Approaches

Comparing Pairwise Approaches

Comparing Pairwise Approaches

- ★ Comparing two radically different approaches to give insights to the tester [Perrouin2012]
  - Alloy-based solution [Perrouin2010]
  - CSP-based solution [Oster2010]

Comparing Pairwise Approaches

- ★ Comparing two radically different approaches to give insights to the tester [Perrouin2012]
  - Alloy-based solution [Perrouin2010]
  - CSP-based solution [Oster2010]

Comparing Pairwise Approaches

- ★ Comparing two radically different approaches to give insights to the tester [Perrouin2012]
  - Alloy-based solution [Perrouin2010]
  - CSP-based solution [Oster2010]

- ★ Conflicting philosophies
  - Generality for Alloy-based
  - Specialization for CSP-based

Key Differences

Key Differences

- ★ FM Expressivity

Key Differences

★ FM Expressivity

	CSP-based	Alloy-based
Cardinalities	-	+
Multiple parents	-	+
Binary constraints	+	+
N-ary constraints	-	+

Key Differences

- ★ FM Expressivity
- ★ Scalability
  - ‘A priori’: Flattening of the FM
  - ‘A posteriori’ : “divide-and-compose” strategies

	CSP-based	Alloy-based
Cardinalities	-	+
Multiple parents	-	+
Binary constraints	+	+
N-ary constraints	-	+

Key Differences

★ FM Expressivity

★ Scalability

- ‘A priori’: Flattening of the FM
- ‘A posteriori’ : “divide-and-compose” strategies

	CSP-based	Alloy-based
Cardinalities	-	+
Multiple parents	-	+
Binary constraints	+	+
N-ary constraints	-	+

★ Determinism

- CSP-based provides always the same suite on a given FM
- Alloy-based can produce very different test suites due to random tuple combinations and scope influence

Experiments

Experiments

- ★ Ran experiments on SPLOT [Mendonca2009]
  - T=[2..3]

Experiments

- ★ Ran experiments on SPLOT [Mendonca2009]
  - T=[2..3]
- ★ Execution times (T=2)

Experiments

- ★ Ran experiments on SPLOT [Mendonca2009]
  - T=[2..3]
- ★ Execution times (T=2)

	CP	SH	AG	MT	ES
Features	19	35	61	88	287
Possible Products	61	1048576	$3.3 * 10^9$	$1.65 * 10^{13}$	$2.26 * 10^{49}$
Cross-Tree Constraints (%)	26	0	55	0	11
CSP-Dedicated (ms)	0	0	32	46	797
BinarySplit (ms)	11812	11457	33954	> 32400000	> 32400000
IncGrowth (ms)	56494	1372094	13847835	> 32400000	> 32400000

Experiments

- ★ Ran experiments on SPLOT [Mendonca2009]
  - T=[2..3]
- ★ Execution times (T=2)

	CP	SH	AG	MT	ES
Features	19	35	61	88	287
Possible Products	61	1048576	$3.3 * 10^9$	$1.65 * 10^{13}$	$2.26 * 10^{49}$
Cross-Tree Constraints (%)	26	0	55	0	11
CSP-Dedicated (ms)	0	0	32	46	797
BinarySplit (ms)	11812	11457	33954	> 32400000	> 32400000
IncGrowth (ms)	56494	1372094	13847835	> 32400000	> 32400000

★ CSP-Dedicated 1000 times faster

Experiments cont’d

Experiments cont'd

- ★ Test Suite Size (T=2)

Experiments cont'd

★ Test Suite Size (T=2)

	CP	SH	AG	MT	ES
CSP-Dedicated	8	40	46	92	215
BinarySplit	12	92	514	N/A	N/A
IncGrowth	15	28	74	N/A	N/A

Experiments cont'd

★ Test Suite Size (T=2)

	CP	SH	AG	MT	ES
CSP-Dedicated	8	40	46	92	215
BinarySplit	12	92	514	N/A	N/A
IncGrowth	15	28	74	N/A	N/A

★ Test Suite Size (T=3)

Experiments cont'd

★ Test Suite Size (T=2)

	CP	SH	AG	MT	ES
CSP-Dedicated	8	40	46	92	215
BinarySplit	12	92	514	N/A	N/A
IncGrowth	15	28	74	N/A	N/A

★ Test Suite Size (T=3)

	CP	SH	AG	MT	ES
CSP-Dedicated	23	61	257	643	841
BinarySplit	207	N/A	N/A	N/A	N/A
IncGrowth	133	N/A	N/A	N/A	N/A

Experiments cont'd

★ Test Suite Size (T=2)

	CP	SH	AG	MT	ES
CSP-Dedicated	8	40	46	92	215
BinarySplit	12	92	514	N/A	N/A
IncGrowth	15	28	74	N/A	N/A

★ Test Suite Size (T=3)

	CP	SH	AG	MT	ES
CSP-Dedicated	23	61	257	643	841
BinarySplit	207	N/A	N/A	N/A	N/A
IncGrowth	133	N/A	N/A	N/A	N/A

★ Observed also increasing numbers of duplicates when T is higher or for larger FM for Alloy-based

Experiments cont'd

★ Test Suite Size (T=2)

	CP	SH	AG	MT	ES
CSP-Dedicated	8	40	46	92	215
BinarySplit	12	92	514	N/A	N/A
IncGrowth	15	28	74	N/A	N/A

★ Test Suite Size (T=3)

	CP	SH	AG	MT	ES
CSP-Dedicated	23	61	257	643	841
BinarySplit	207	N/A	N/A	N/A	N/A
IncGrowth	133	N/A	N/A	N/A	N/A

★ Observed also increasing numbers of duplicates when T is higher or for larger FM for Alloy-based

★ There is a clear winner :)

Also...

Also…

★ Other approaches emerged

- Pacogen [Hervieu2011]
- SPLCAT [Johansen2011,2012a,2012b]
- Search-based techniques
  - GA + Fitness: T-wise coverage [Ensan2012]
  - GA + Fitness: Similarity: http://research.henard.net/SPL/

Also…

★ Other approaches emerged

- Pacogen [Hervieu2011]
- SPLCAT [Johansen2011,2012a,2012b]
- Search-based techniques
  - GA + Fitness: T-wise coverage [Ensan2012]
  - GA + Fitness: Similarity: http://research.henard.net/SPL/

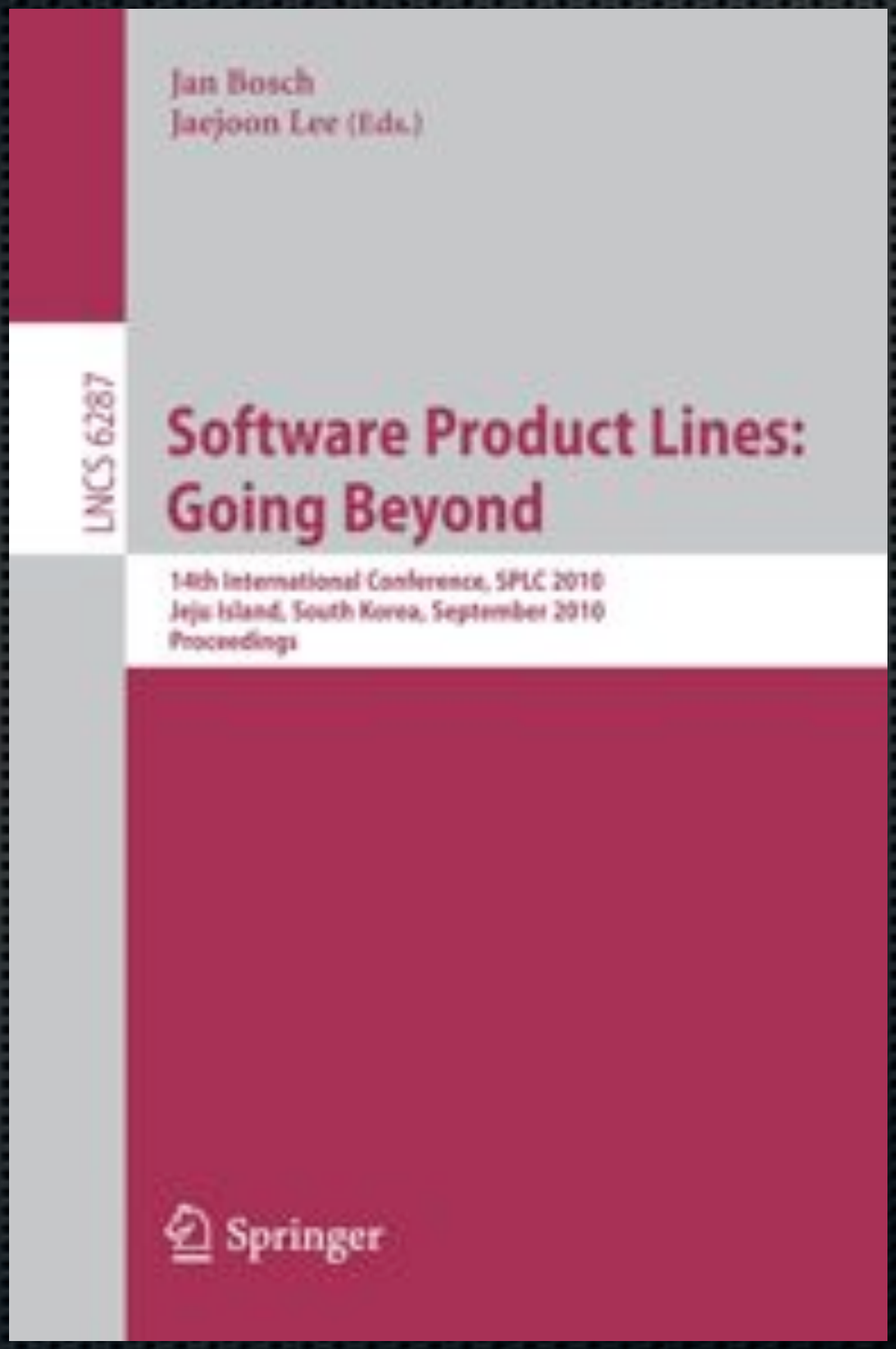
★ Scalability greatly improved

- From dozens to thousands of features (linux FM)

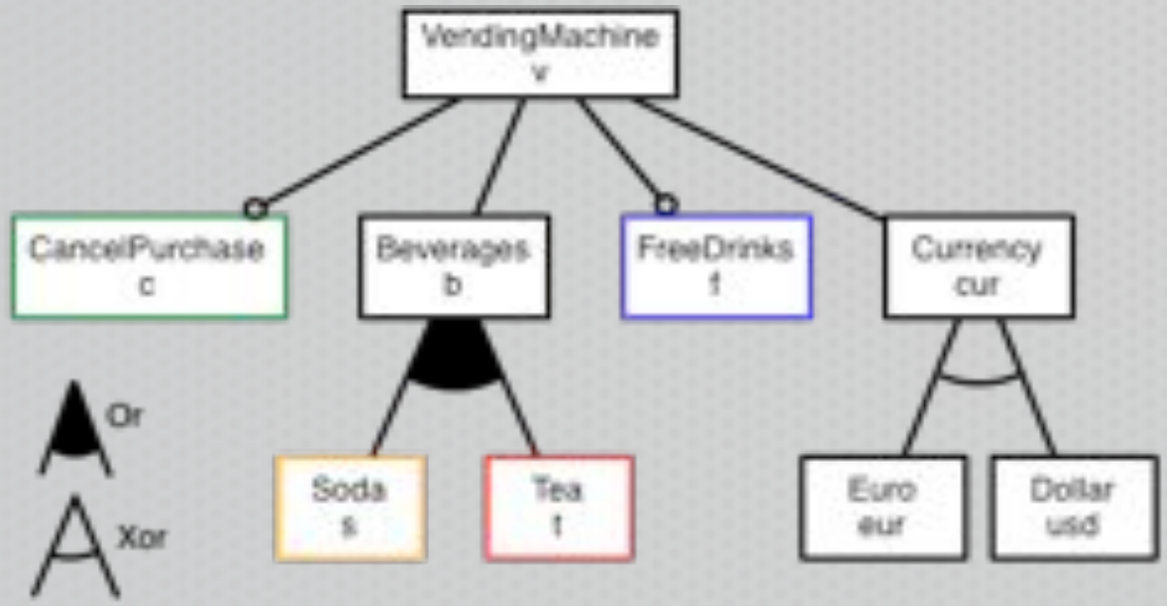
Also…

- ★ Other approaches emerged
  - Pacogen [Hervieu2011]
  - SPLCAT [Johansen2011,2012a,2012b]
  - Search-based techniques
    - GA + Fitness: T-wise coverage [Ensan2012]
    - GA + Fitness: Similarity: http://research.henard.net/SPL/
- ★ Scalability greatly improved
  - From dozens to thousands of features (linux FM)
- ★ T-wise is “blind” => new challenges
  - Prioritization, Non-boolean feature models...

Beyond pairwise....

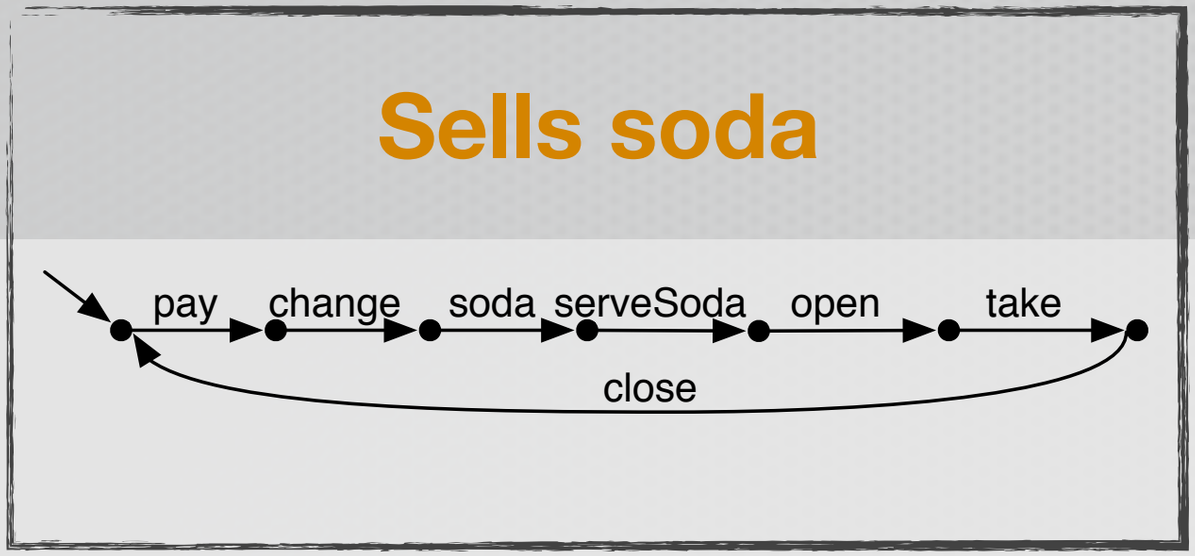


Unified Behavioural SPL QA



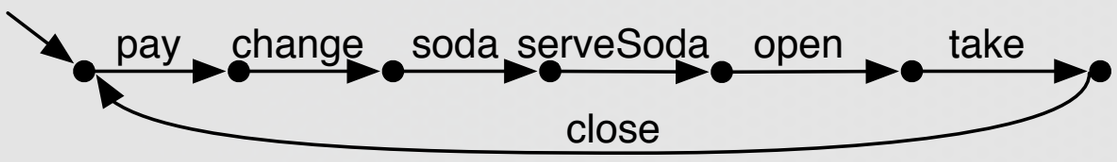
Featured Transitions Systems

Featured Transitions Systems

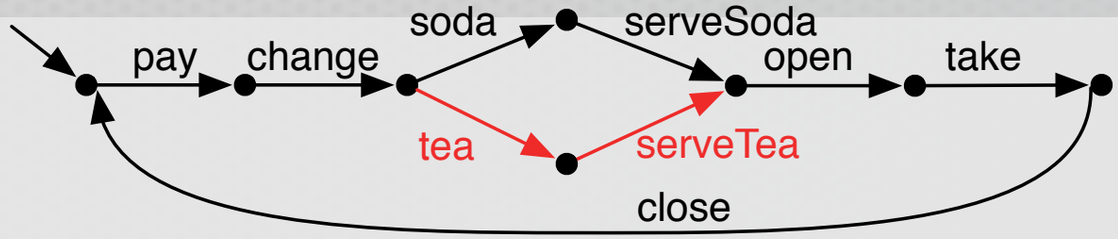


Featured Transitions Systems

Sells soda

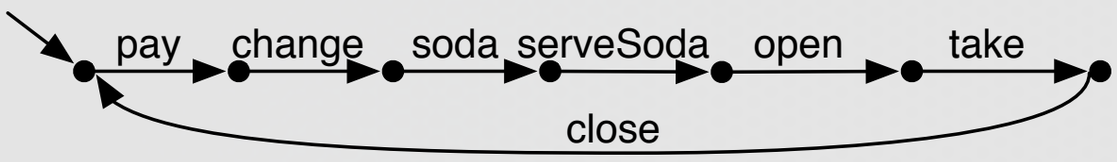


Sells soda and tea

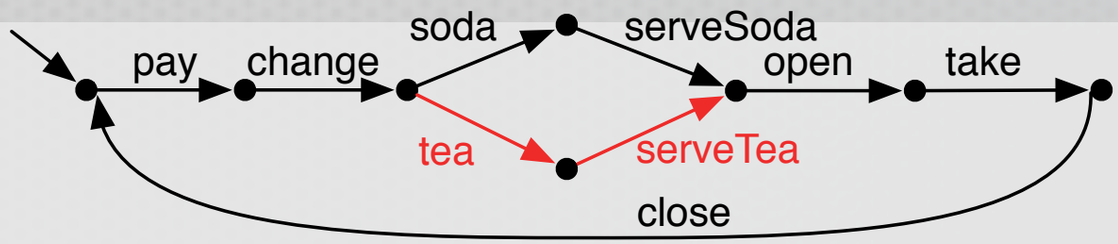


Featured Transitions Systems

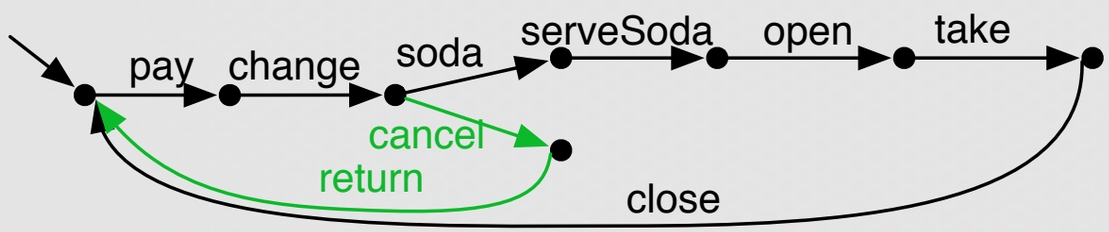
Sells soda



Sells soda and tea

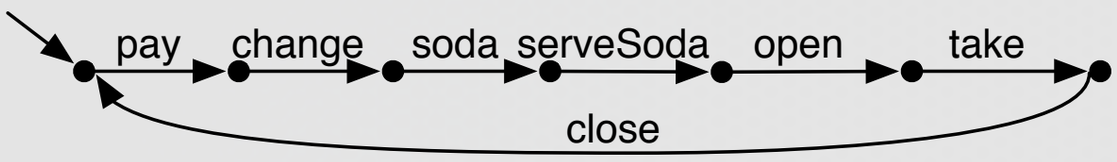


Can cancel purchase

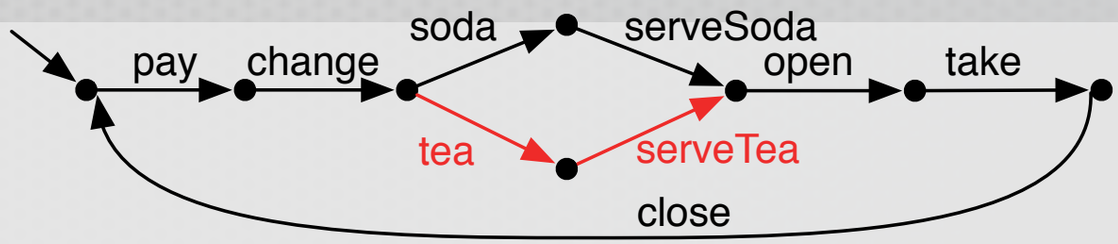


Featured Transitions Systems

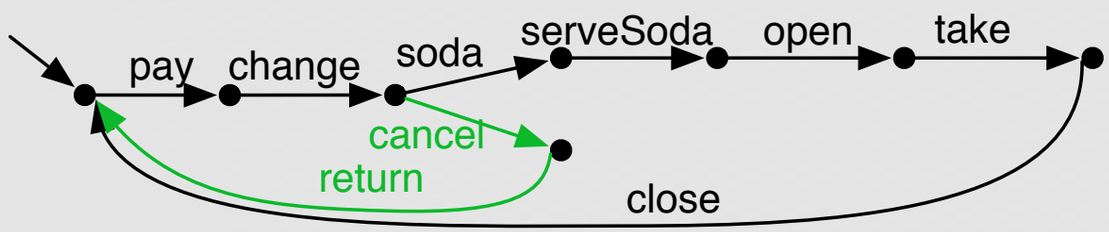
Sells soda



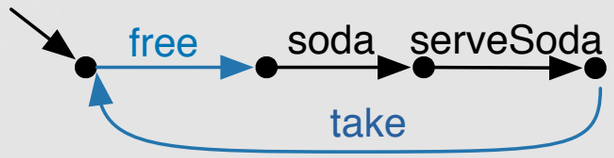
Sells soda and tea



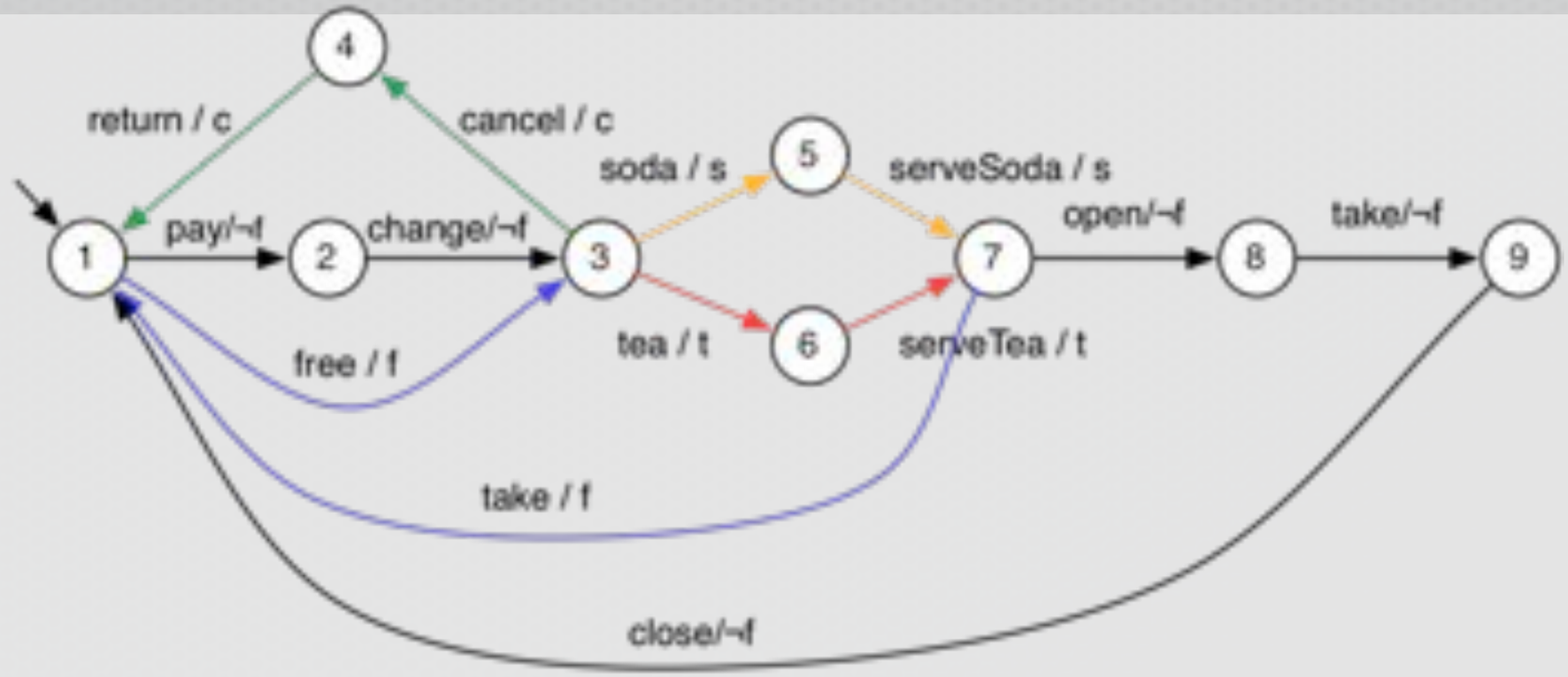
Can cancel purchase



Drinks are free



Featured Transitions Systems



FTS cont’d

FTS cont'd

- ★ Designed for Model-Checking
  - Exponentially more efficient than product-by-product verification
  - Tool-support: SNIP [Classen2012], NuSMV [Classen2011]
  - Real-time [Cordy2012a], adaptive systems [Cordy2012b]

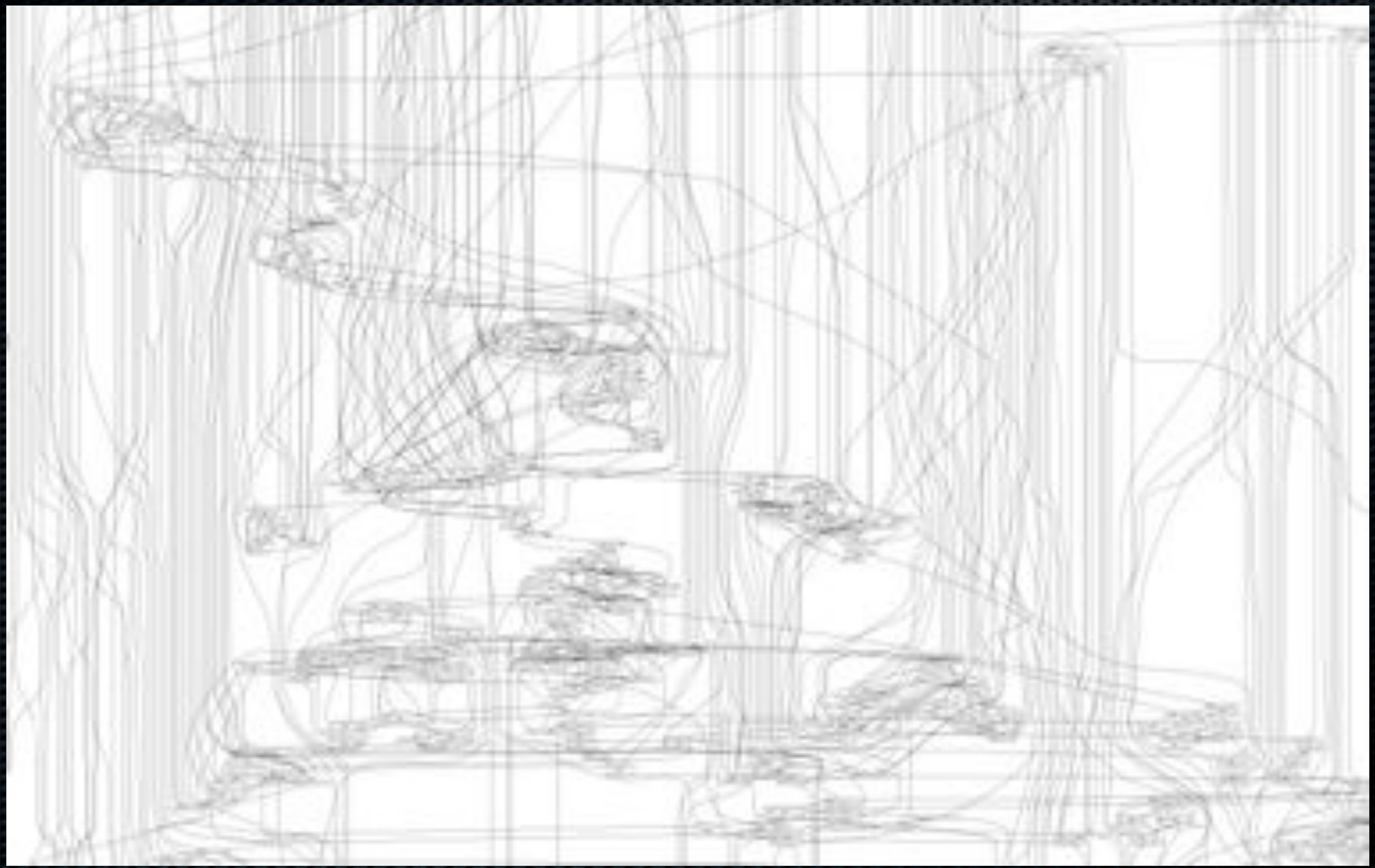
FTS cont'd

- ★ Designed for Model-Checking
  - Exponentially more efficient than product-by-product verification
  - Tool-support: SNIP [Classen2012], NuSMV [Classen2011]
  - Real-time [Cordy2012a], adaptive systems [Cordy2012b]
- ★ SPL-dedicated
  - From product to set of products
  - From application to domain engineering

FTS cont'd

- ★ Designed for Model-Checking
  - Exponentially more efficient than product-by-product verification
  - Tool-support: SNIP [Classen2012], NuSMV [Classen2011]
  - Real-time [Cordy2012a], adaptive systems [Cordy2012b]
- ★ SPL-dedicated
  - From product to set of products
  - From application to domain engineering
- ★ Goal: Combination with Testing
  - MC properties as test selection criteria
  - Verification of feature interactions

Leveraging FTS



Leveraging FTS

Leveraging FTS

- ★ Not really an user-friendly language
  - No structuring mechanism
  - Higher-level models (fPromela, fSMV) still requires MC expertise

Leveraging FTS

- ★ Not really an user-friendly language
  - No structuring mechanism
  - Higher-level models (fPromela, fSMV) still requires MC expertise
- ★ Use of UML instead
  - Broaden the scope of this techniques to any SPL engineer
  - Abstraction: Hierarchical states, orthogonal regions
  - FTS as underlying formal semantics

Leveraging FTS

- ★ Not really an user-friendly language
  - No structuring mechanism
  - Higher-level models (fPromela, fSMV) still requires MC expertise
- ★ Use of UML instead
  - Broaden the scope of this techniques to any SPL engineer
  - Abstraction: Hierarchical states, orthogonal regions
  - FTS as underlying formal semantics
- ★ Challenges
  - UML 2 FTS : flattening...
  - Testable FTS: Extended Actions, test criteria, FTS-ioco...

Conclusions

Conclusions

- ★ SPL Testing was ignored for long, but...
  - Gaining momentum
  - Huge progress in applicability and scalability for T-wise techniques => ready for industry ?

Conclusions

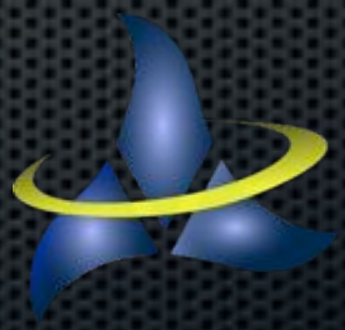
- ★ SPL Testing was ignored for long, but...
  - Gaining momentum
  - Huge progress in applicability and scalability for T-wise techniques => ready for industry ?
- ★ T-wise is “blind”
  - prioritisation (weights, ordered suites...)
  - flexibility (time/budget constraints)

Conclusions

- ★ SPL Testing was ignored for long, but...
  - Gaining momentum
  - Huge progress in applicability and scalability for T-wise techniques => ready for industry ?
- ★ T-wise is “blind”
  - prioritisation (weights, ordered suites...)
  - flexibility (time/budget constraints)
- ★ Move to behavioural SPL Testing & QA
  - Tough Challenge: collaboration between Verification and Testing communities required

Thanks...

- ★ Benoit Baudry, Sagar Sen, Jacques Klein, Yves Le Traon, Sebastian Oster
- ★ Arnaud Gotlieb, Aymeric Hervieu
- ★ Xavier Devroey, Maxime Cordy, Patrick Heymans, Pierre-Yves Schobbens, Axel Legay, Eun-Young Kang, Andreas Classen
- ★ Christopher Hénard, Mike Papadakis



References

- ★ [Asirelli2001] Asirelli, P., ter Beek, M.H., Fantechi, A., Gnesi, S., Mazzanti, F.: Design and validation of variability in product lines. In: Proceedings of the 2nd International Workshop on Product Line Approaches in Software Engineering. pp. 25–30. PLEASE ’11, ACM, New York, NY, USA (2011)
- ★ [Classen2010] Classen, A., Heymans, P., Schobbens, P., Legay, A., Raskin, J.: Model checking lots of systems: efficient verification of temporal properties in software product lines. In: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1. pp. 335–344. ICSE ’10, ACM, New York, NY, USA (2010)
- ★ [Classen2011] Classen, A., Heymans, P., Schobbens, P., Legay, A.: Symbolic model checking of software product lines. In: Proceedings 33rd International Conference on Software Engineering (ICSE 2011). ACM Press, New York (2011)
- ★ [Fischbein2006] Fischbein,D.,Uchitel,S.,Braberman,V.:Afoundationforbehaviouralconformanceinsoft- ware product line architectures. In: Proceedings of the ISSTA 2006 workshop on Role of software architecture for testing and analysis. pp. 39–48. ROSATEA ’06, ACM, New York, NY, USA (2006)
- ★ [Gruler2008] Gruler, A., Leucker, M., Scheidemann, K.: Modeling and model checking software product lines. In: Barthe, G., Boer, F.S. (eds.) Formal Methods for Open Object-Based Distributed Systems. vol. 5051, pp. 113–131. Springer-Verlag, Berlin, Heidelberg (2008)
- ★ [Lauenroth2009] Lauenroth,K.,Pohl,K.,Toehning,S.:Model checking of domain artifacts in productline engineering. In: Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering. pp. 269–280. ASE ’09, IEEE Computer Society, Washington, DC, USA (2009)
- ★ [Li2002] Li, H.C., Krishnamurthi, S., Fisler, K.: Interfaces for modular feature verification. In: Pro- ceedings of the 17th IEEE international conference on Automated software engineering. pp. 195–204. ASE ’02, IEEE Computer Society, Washington, DC, USA (2002)

References

- ★ [Oster et al 2010] Sebastian Oster, Florian Markert, Philipp Ritter: Automated Incremental Pairwise Testing of Software Product Lines. SPLC 2010:196-210
- ★ [Perrouin2008] Gilles Perrouin, Jacques Klein, Nicolas Guelfi, Jean-Marc Jézéquel: Reconciling Automation and Flexibility in Product Derivation. SPLC 2008: 339-348
- ★ [Perrouin2010] Gilles Perrouin, Sagar Sen, Jacques Klein, Benoit Baudry, Yves Le Traon: Automated and Scalable T-wise Test Case Generation Strategies for Software Product Lines. ICST 2010: 459-468
- ★ [Perrouin2012] Gilles Perrouin, Sebastian Oster, Sagar Sen, Jacques Klein, Benoit Baudry, Yves Le Traon: Pairwise testing for software product lines: comparison of two approaches. Software Quality Journal 20(3-4): 605-643 (2012)
- ★ [Uzu08] E. Uzuncaova, D. Garcia, S. Khurshid, and D. Batory, “Testing software product lines using incremental test generation,” in ISSRE. IEEE Computer Society, 2008, pp. 249–258.
- ★ [Cohen2006] M. B. Cohen, M. B. Dwyer, and J. Shi, “Coverage and adequacy in software product line testing,” in ROSATEA@ISSTA, 2006, pp. 53–63.[10]
- ★ [Cohen2007] M. Cohen, M. Dwyer, and J. Shi, “Interaction testing of highly-configurable systems in the presence of constraints,” in ISSTA, 2007, pp. 129–139.
- ★ [Weißleder2010] Stephan Weißleder: Test models and coverage criteria for automatic model-based test generation with UML state machines. PhD Thesis, Humboldt University of Berlin 2010, pp. 1-259
- ★ [Utting2006] Utting,M.,Legeard,B.:Practicalmodel-based testing: a tools approach. Morgan Kaufmann, 2006
- ★ [Kuhn2004] Kuhn DR, Wallace DR, Gallo AM (2004) Software fault interactions and implications for software testing. IEEE Trans Softw Eng 30(6):418–421

References

- ★ [Batory2005] D. S. Batory, “Feature models, grammars, and propositional formulas,”in SPLC, 2005, pp. 7–20.
- ★ [Czarnecki2007] K. Czarnecki and A. Wasowski, “Feature diagrams and logics: There and back again,” in SPLC.Los Alamitos, CA, USA: IEEE ComputerSociety, 2007, pp. 23–34.
- ★ [Schobbens2007] P. Schobbens, P. Heymans, J. Trigaux, and Y. Bontemps, “Generic semantics of feature diagrams,” Computer Networks, vol. 51, no. 2, pp.456–479, 2007.
- ★ [Benavides2010] Benavides D, Segura S, Ruiz-Cortés A (2010) Automated analysis of feature models 20 years later: A literature review. Information Systems 35(6):615 – 63
- ★ [Mendonca2009] Mendonca M, Branco M, Cowan D (2009) SPLOT: software product lines online tools. In: Proceeding of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications, ACM, pp 761–762
- ★ [Hervieu2011] Aymeric Hervieu, Benoit Baudry, Arnaud Gotlieb: PACOGEN: Automatic Generation of Pairwise Test Configurations from Feature Models. ISSRE 2011: 120-129
- ★ [Johansen2012a] Martin Fagereng Johansen, Øystein Haugen, Franck Fleurey, Anne Grete Eldegard, Torbjørn Syversen: Generating Better Partial Covering Arrays by Modeling Weights on Sub-product Lines. MoDELS 2012: 269-284
- ★ [Johansen2012b] Martin Fagereng Johansen, Øystein Haugen, Franck Fleurey: An algorithm for generating t-wise covering arrays from large feature models. SPLC (1) 2012: 46-55
- ★ [Johansen2011] Martin Fagereng Johansen, Øystein Haugen, Franck Fleurey: Properties of Realistic Feature Models Make Combinatorial Testing of Product Lines Feasible. MoDELS 2011: 638-652

References

- ★ [Classen2012] Classen, A.; Cordy, M.; Heymans, P.; Legay, A. and Schobbens, P-Y. Model checking software product lines with SNIP. In International Journal on Software Tools for Technology Transfer (STTT), Springer-Verlag, 14 (5): 589-612, 2012.
- ★ [Cordy2012a] Maxime Cordy, Pierre-Yves Schobbens, Patrick Heymans, Axel Legay: Behavioural modelling and verification of real-time software product lines. SPLC (1) 2012: 66-75
- ★ [Cordy2012b] Maxime Cordy, Andreas Classen, Patrick Heymans, Axel Legay, Pierre-Yves Schobbens. Model Checking Adaptive Software with Featured Transition Systems, in Assurance for Self-Adaptive Systems, Lecture Notes in Computer Science, to appear.
- ★ [Classen2008] Classen A, Heymans P, Schobbens P (2008) What's in a feature: A require- ments engineering perspective. In: Proceedings of the Theory and practice of software, 11th international conference on Fundamental approaches to software engineering, Springer-Verlag, pp 16–30
- ★ [Classen2010b] Andreas Classen, Quentin Boucher, Patrick Heymans: A text-based approach to feature modelling: Syntax and semantics of TVL. Sci. Comput. Program. 76(12): 1130-1143 (2011)