Extending Dynamic Software Product Lines with Temporal Constraints

Gustavo Sousa Walter Rudametkin Laurence Duchien

firstname.lastname@inria.fr

SEAMS 2017

(12th Int. Symp. on Soft. Eng. for Adaptive and Self-Managing Systems)

Adaptive Cloud Environments

- Cloud computing supports construction of customized adaptable environments

> "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) **that can be rapidly provisioned and released with minimal management effort or service provider interaction**."[1]

- A cloud environment is a set of cloud services provisioned for running an application

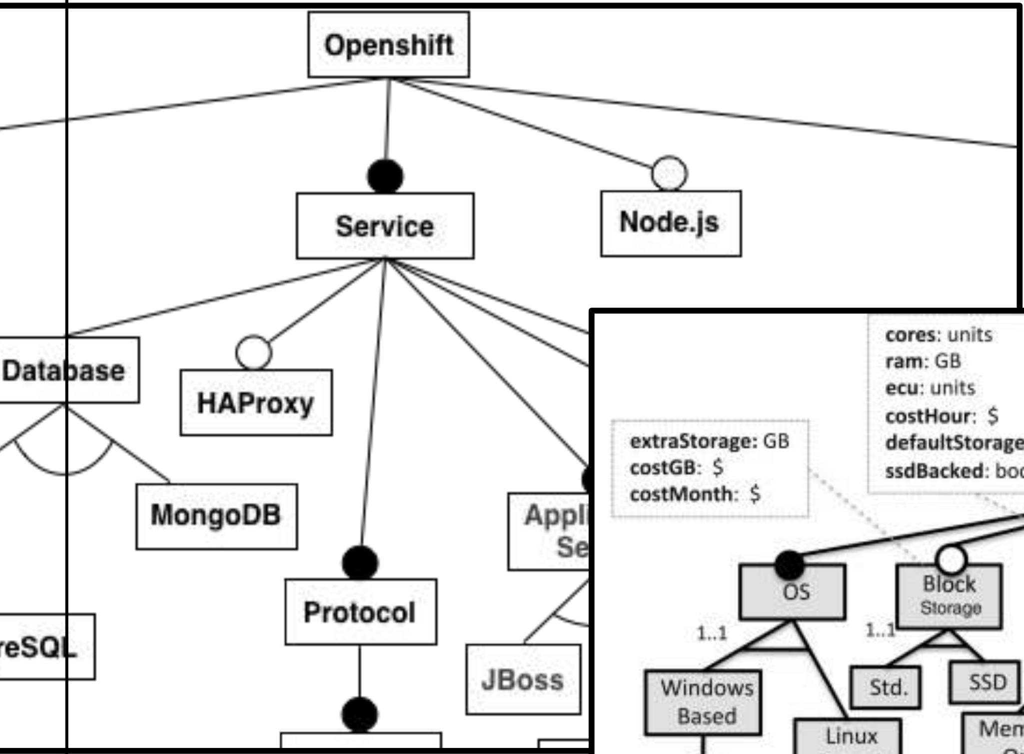
[1] P. Mell and T. Grance, "The NIST definition of cloud computing," Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, Tech. Rep., 2011.

Cloud Providers Configuration Variability

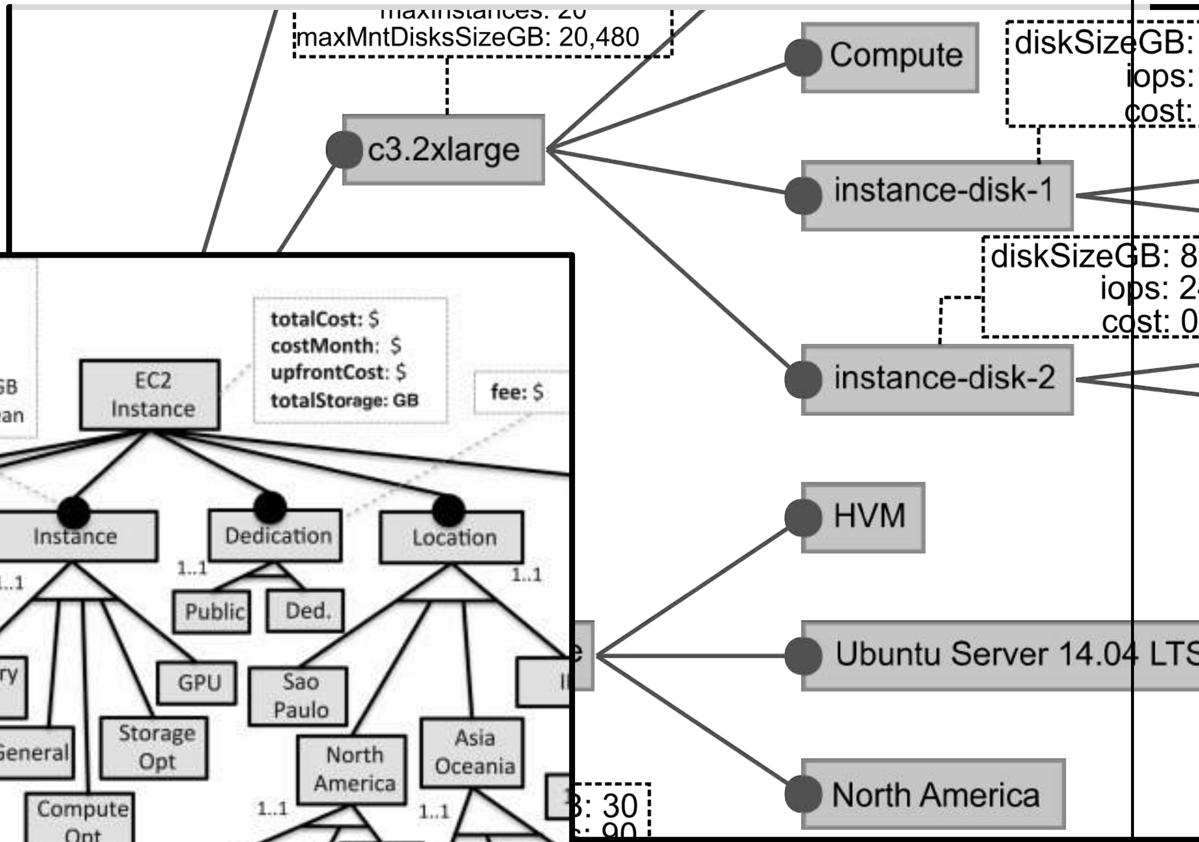
- Wide range of configurable cloud services
- Complex configuration rules and constraints

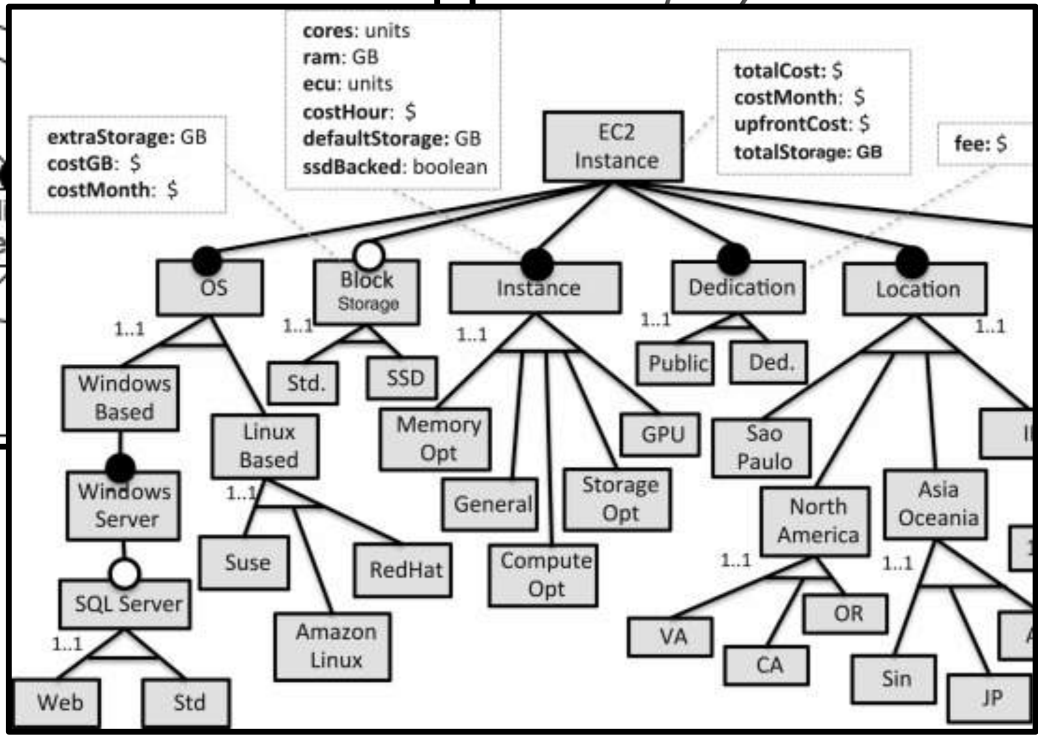
SPLs for Automated Cloud Configuration

C. Quinton et al. (2016) SALOON: a platform for selecting and configuring cloud environments.



A. Ferreira Leite et al. (2015) Automating Resource Selection and Configuration in Inter-clouds through a SPL method.





J. García-Galán et al. (2016) Automated Configuration Support for Infrastructure Migration to the Cloud.

Dynamic Software Product Lines

- **High variability** with **adaptive capabilities**

Dynamic Software Product Lines

- **High variability** with **adaptive capabilities**

- DSPL vs SPL
  - Features can be (re)bound at runtime
  - Adaptive system vs systems family
  - Variability model central to both

Dynamic Software Product Lines

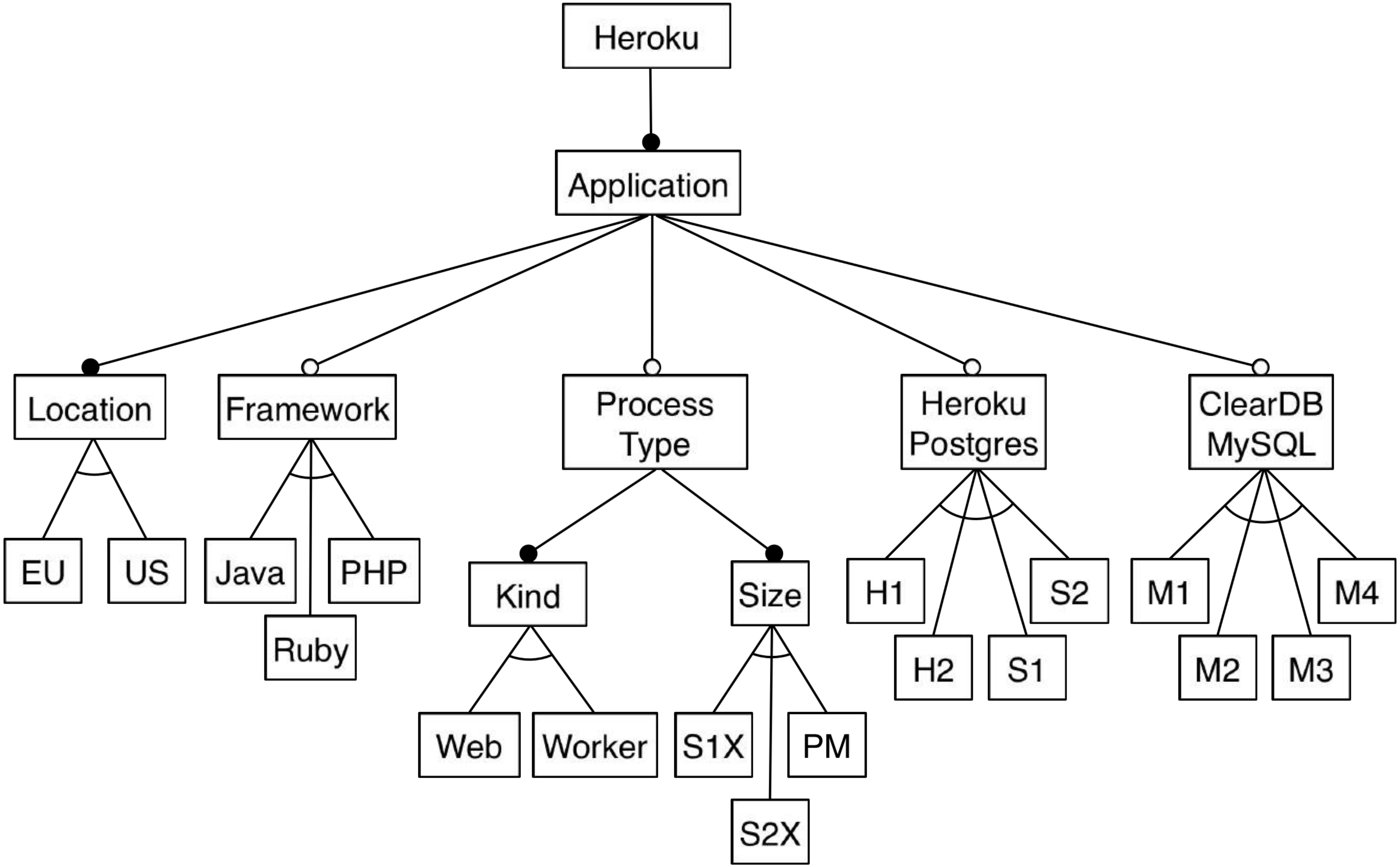
- **High variability** with **adaptive capabilities**
- DSPL vs SPL
  - Features can be (re)bound at runtime
  - Adaptive system vs systems family
  - Variability model central to both
- Adaptation in DSPLs
  - A context change is mapped to a request to include or exclude a set of features from the current configuration
  - SPL analysis is used to derive valid configurations

Cloud Computing Environment

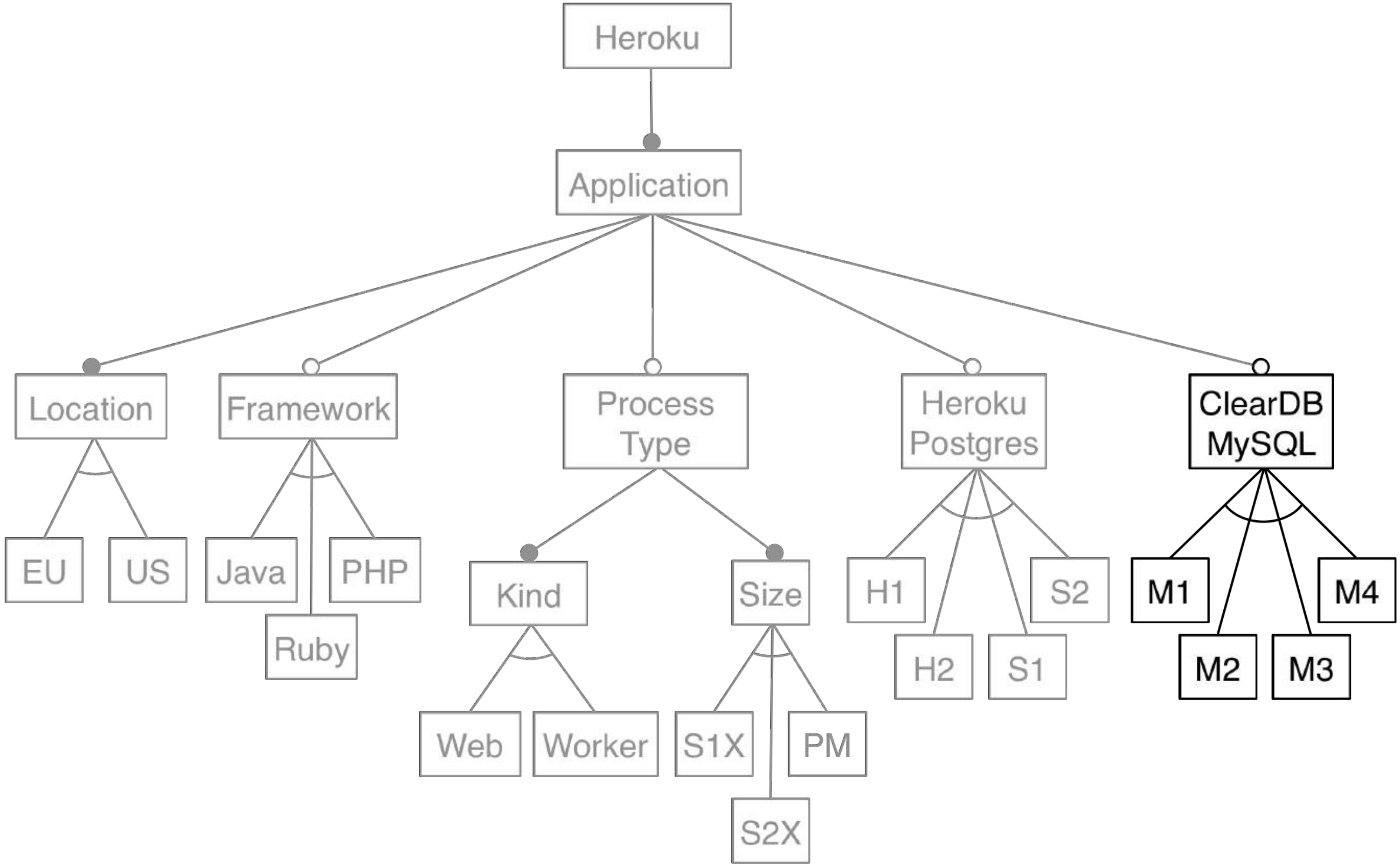
- Reconfiguration mechanisms are provider-dependent and heterogeneous
  - May depend on initial or previous configurations
  - Alternative ways to reconfigure

- Compliance to variability model is not enough
  - Does not ensure valid and safe reconfigurations

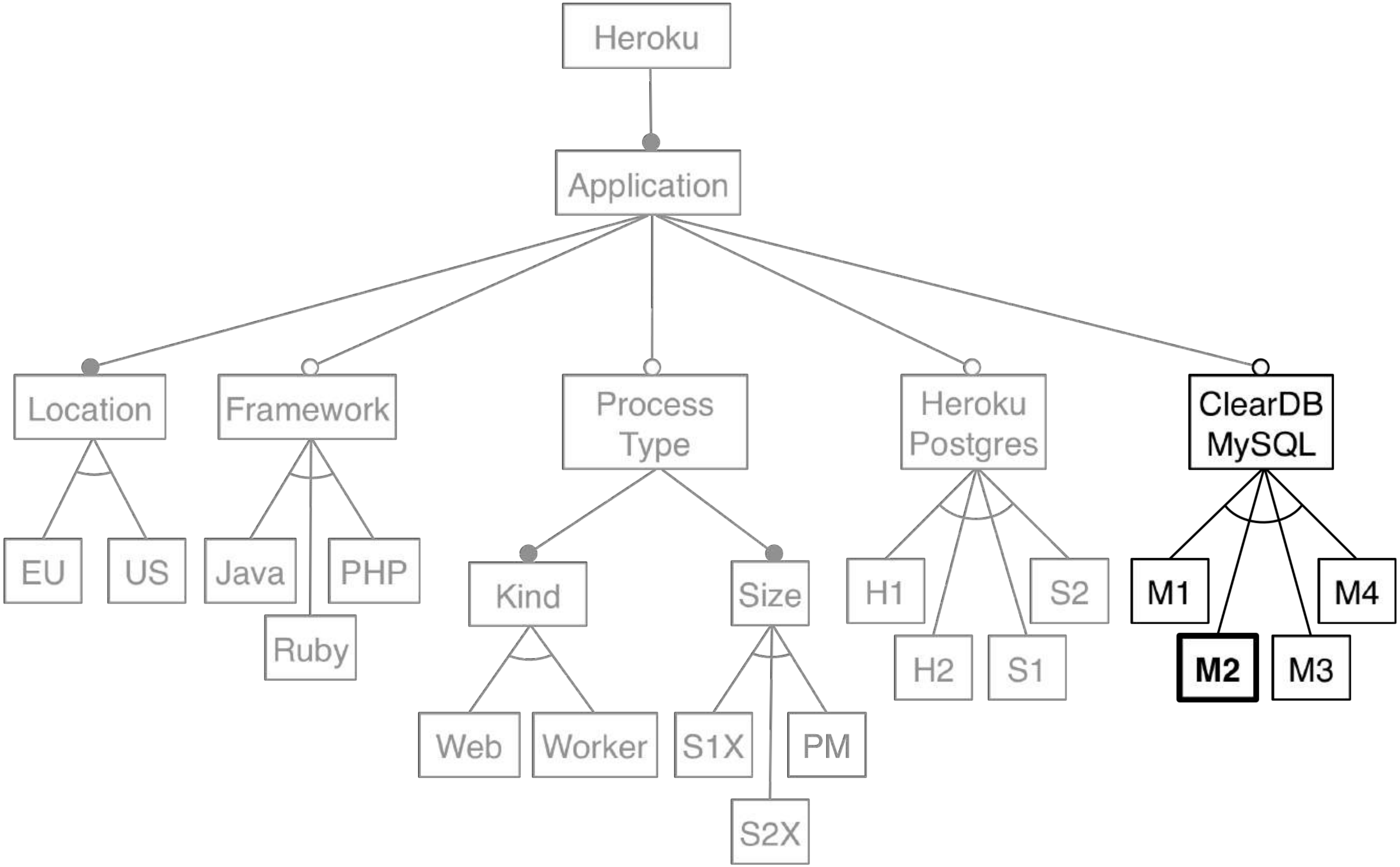
Motivating Example



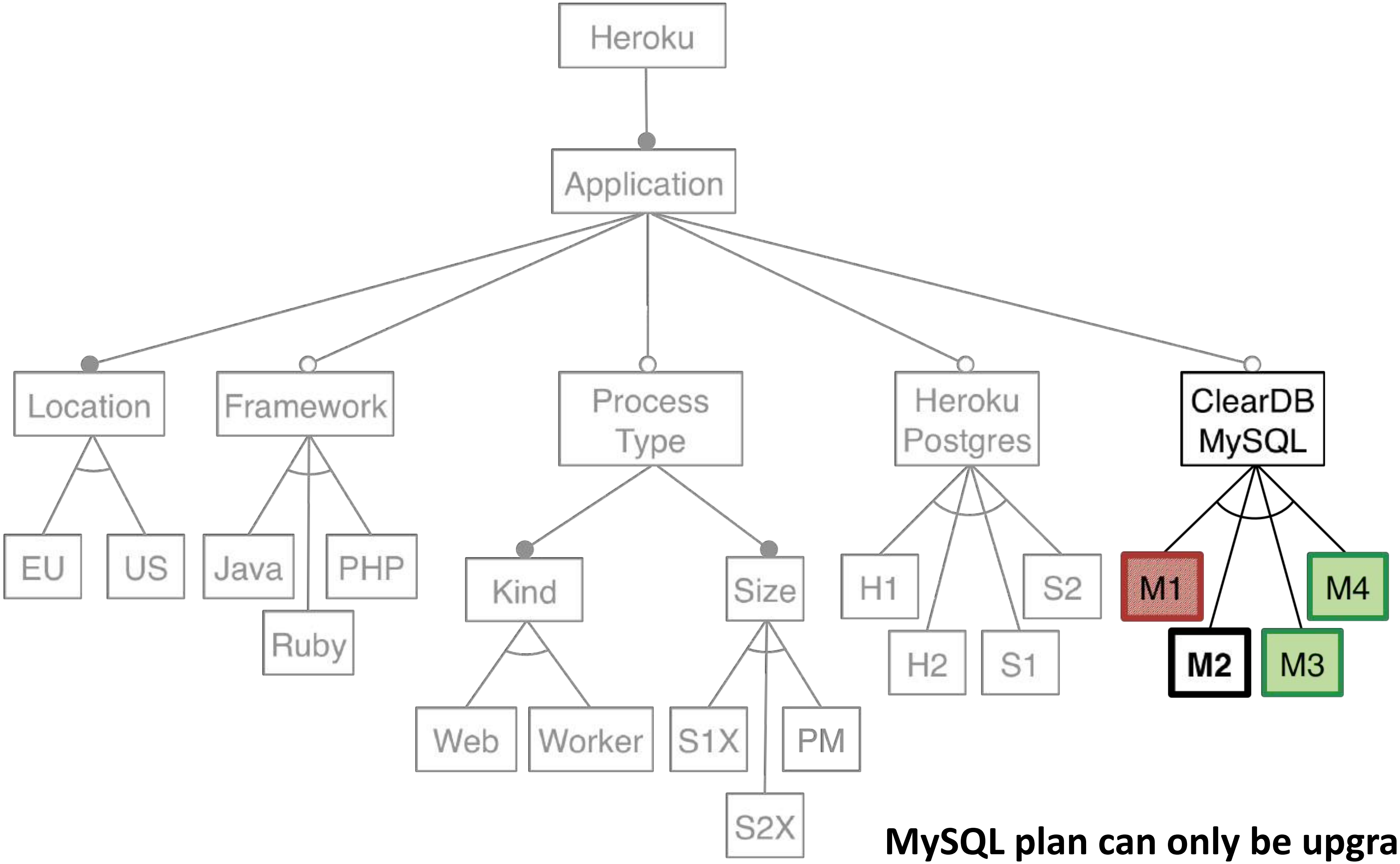
Motivating Example



Motivating Example



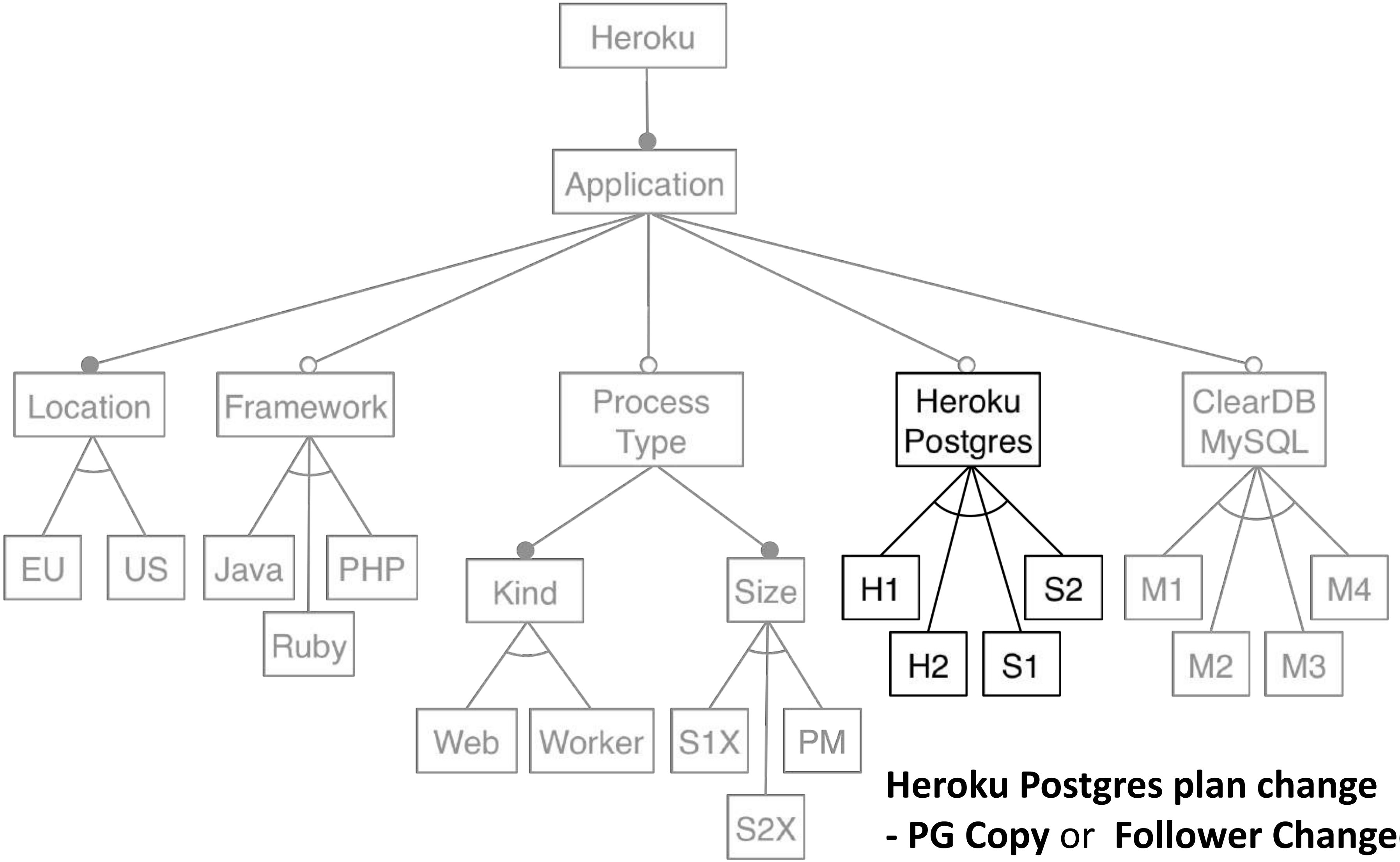
Motivating Example



MySQL plan can only be upgraded

https://devcenter.heroku.com/articles/cleardb#upgrading-your-cleardb-database

Motivating Example

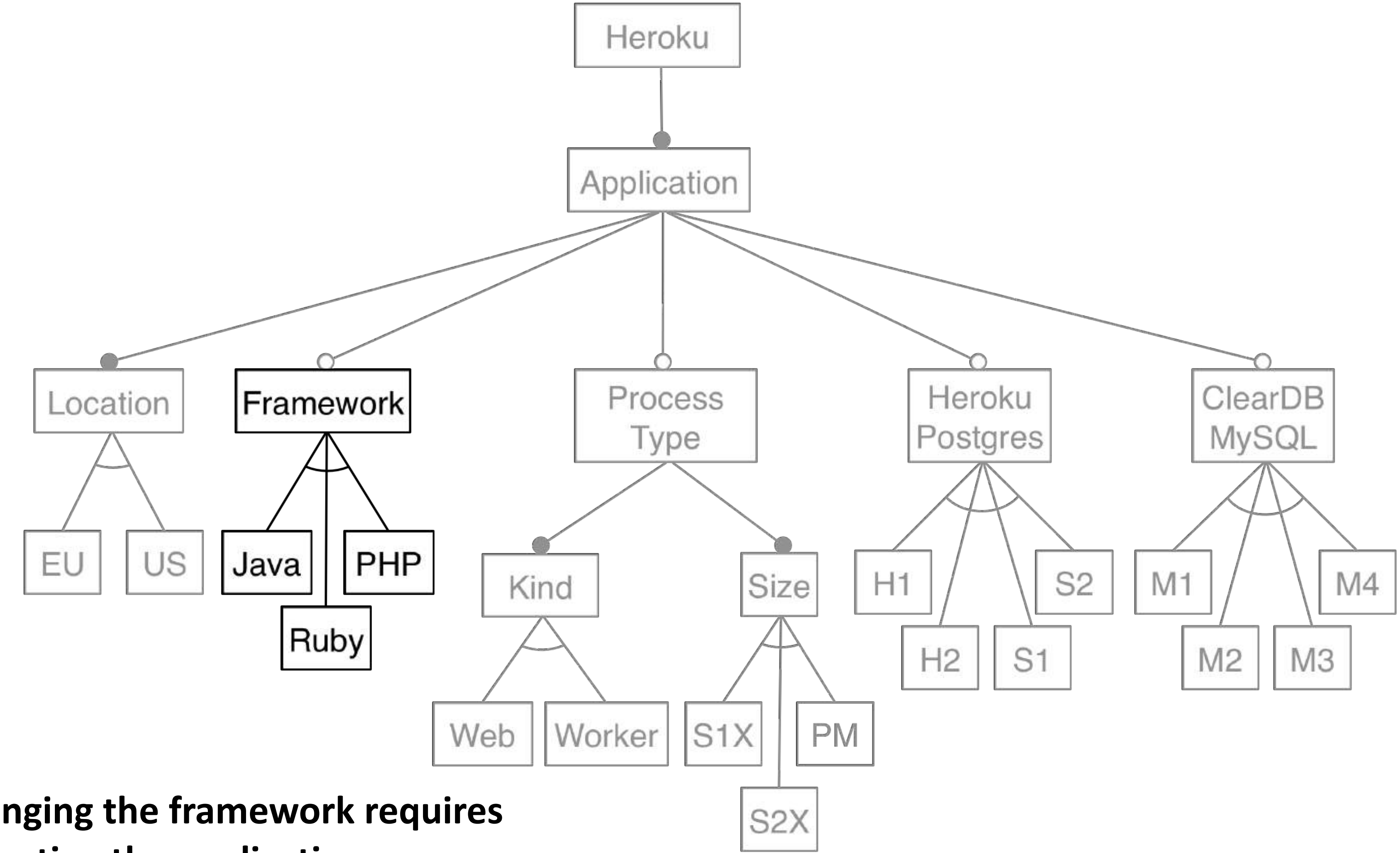


Heroku Postgres plan change

- PG Copy or **Follower Changeover**

https://devcenter.heroku.com/articles/upgrading-heroku-postgres-databases

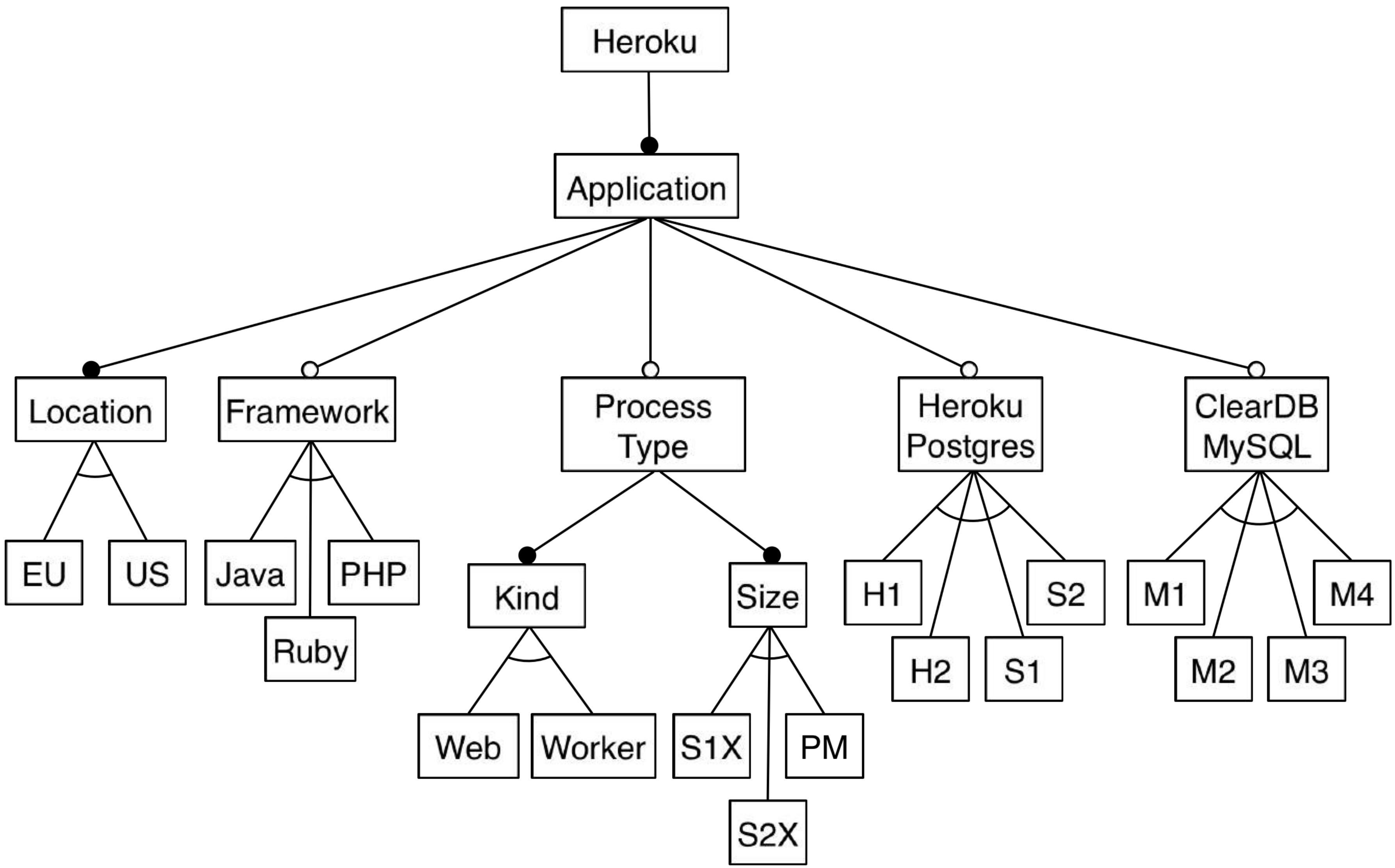
Motivating Example



Changing the framework requires restarting the application

https://devcenter.heroku.com/articles/buildpacks#setting-a-buildpack-on-an-application

Motivating Example



Limitations in DSPLs

- Seminal works on DSPLs highlight the need for validating transitions between system configurations
  - systems should evolve through safe migration paths[6]
  - dynamic constraints on allowed transitions must be considered[7]

- Validation is mostly limited to compliance to a variability model

[6] B. Morin, O. Barais, J. M. Jezequel, F. Fleurey, and A. Solberg, "Models@run.time to support dynamic adaptation," Computer, vol. 42, no. 10, pp. 44–51, Oct 2009.

[7] A. Hubaux and P. Heymans, "On the evaluation and improvement of feature-based configuration techniques in software product lines," in Proc. 31st Int. Conf. Software Engineering (ICSE'09), Vancouver, Canada, May 2009, pp. 367–370.

Problem statement

- How to model constraints over the adaptation behavior?
  - Temporal dependencies between features and reconfiguration operations
- How to reason over a variability model with reconfiguration constraints to find reconfigurations that meet a given criteria?
  - e.g. reduced downtime or costs

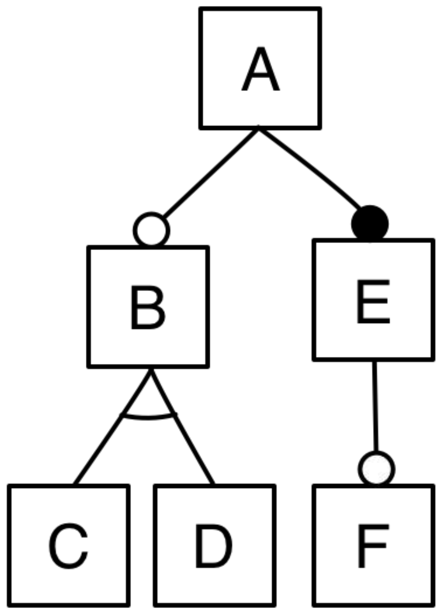
Proposed approach

- Combine **variability models** with **temporal constraints** and **reconfiguration operations**
  - Leverage concepts and solutions from model checking

se

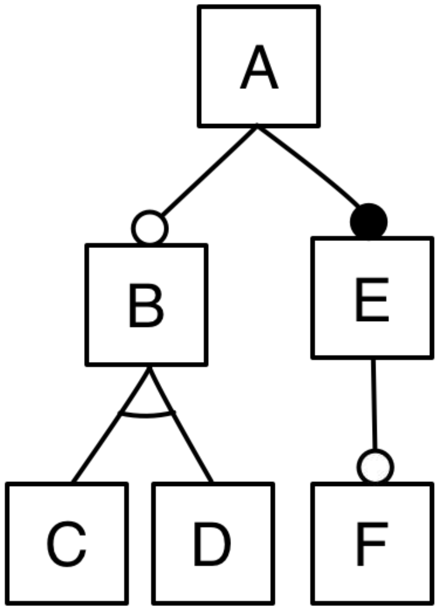
Feature Models and Transition Systems

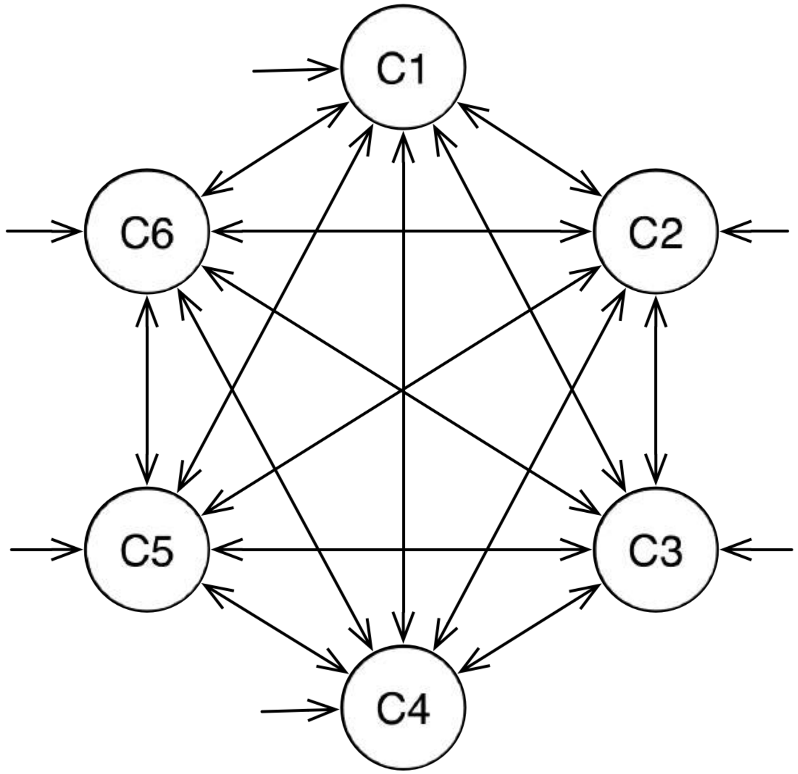
- Feature model $M = (F, C)$
  - F is the set of features
  - $C \subseteq \mathcal{P}(F)$



C1 = {A, E}
C2 = {A, E, F}
C3 = {A, B, C, E}
C4 = {A, B, C, E, F}
C5 = {A, B, D, E}
C6 = {A, B, D, E, F}

DSPLs as Transition Systems



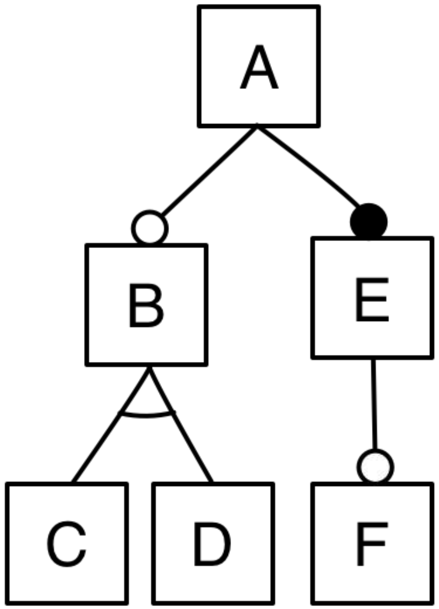


C1 = {A, E}
C2 = {A, E, F}
C3 = {A, B, C, E}
C4 = {A, B, C, E, F}
C5 = {A, B, D, E}
C6 = {A, B, D, E, F}

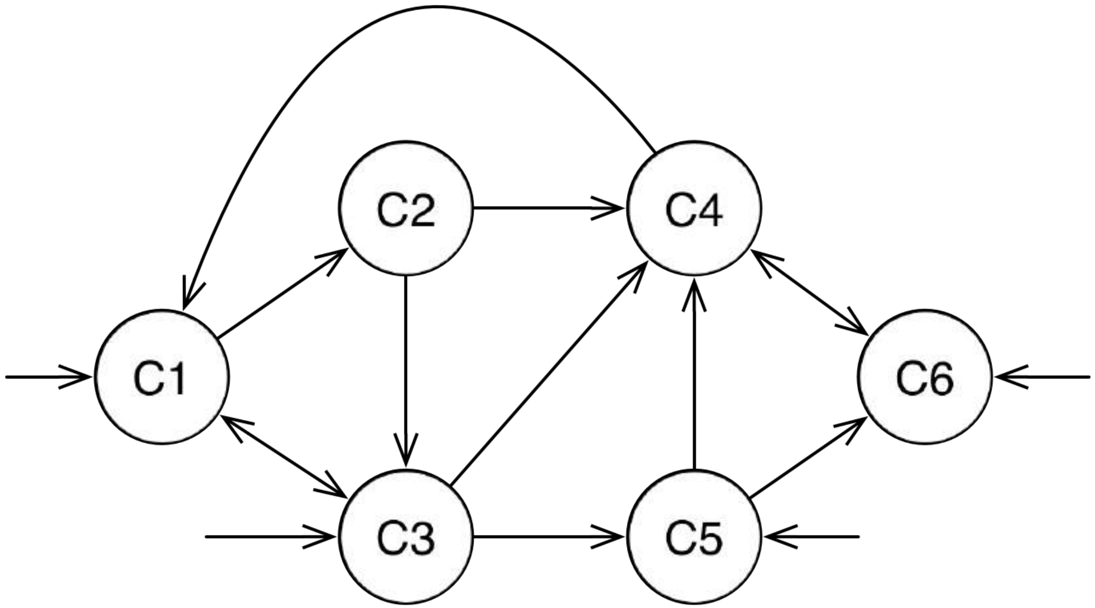
"A DSPL's execution can be abstracted as a highly connected state machine where the states are the possible system configurations and the transitions the migration paths."[6]

[6] B. Morin, O. Barais, J. M. Jezequel, F. Fleurey, and A. Solberg, "Models@run.time to support dynamic adaptation," Computer, vol. 42, no. 10, pp. 44–51, Oct 2009.

DSPLs as Transition Systems



C1 = {A, E}
C2 = {A, E, F}
C3 = {A, B, C, E}
C4 = {A, B, C, E, F}
C5 = {A, B, D, E}
C6 = {A, B, D, E, F}



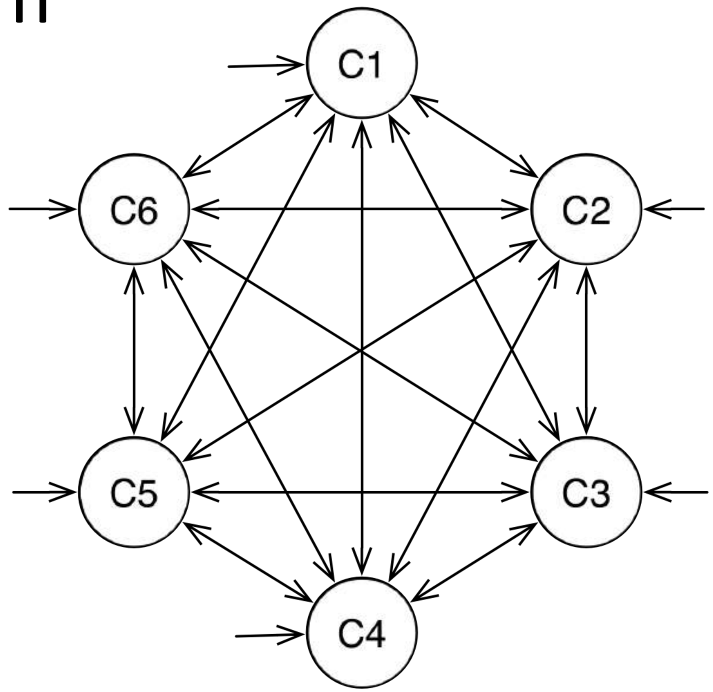
Temporal properties

- A temporal property defines a condition over the executions of a transition system
  - Execution:

    $\rho = s_0 s_1 s_2 s_3 s_4 \ldots$

    $s_i \rightarrow s_{i+1}$ is a transition

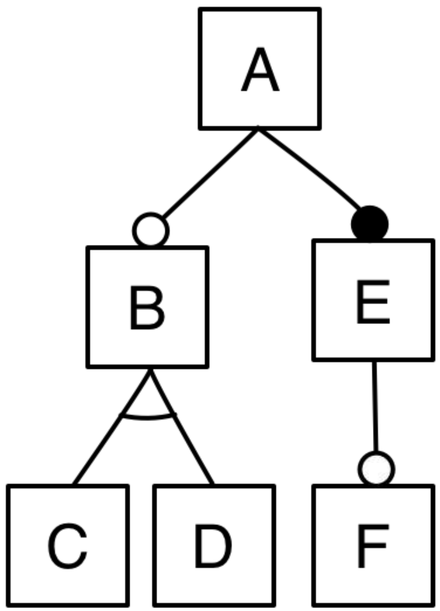
  - A property is a set executions



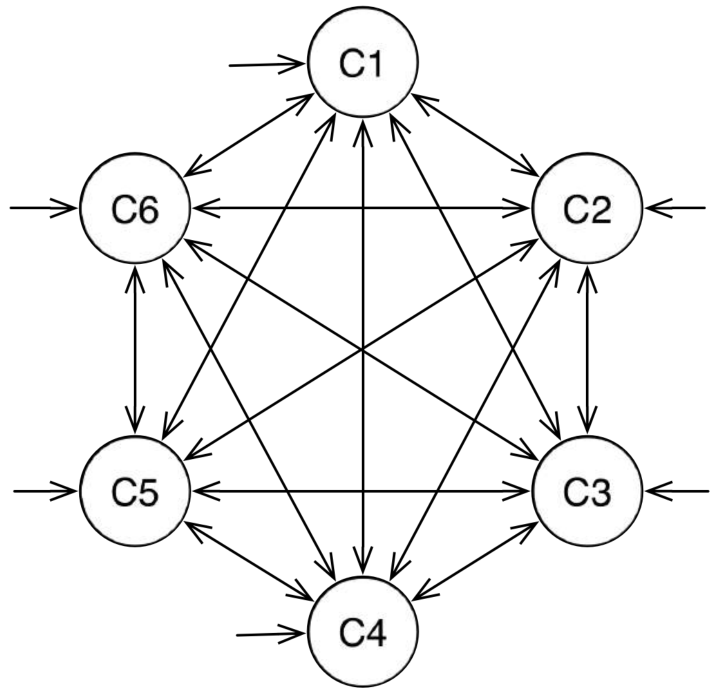
  - A system exhibits a property if all its executions are part of the property set

Feature Models and Transition Systems

- Feature model $M = (F, C)$
- Transition system $TS_M = (S, I, R, AP, L)$
  - $S = I = C$, $R = S \times S$, $AP = F$, $L(x) = x$

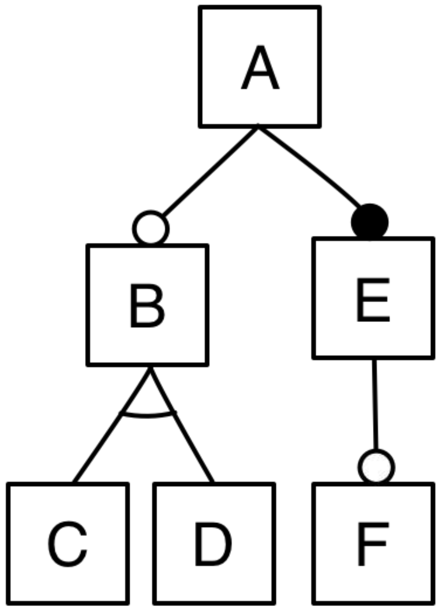


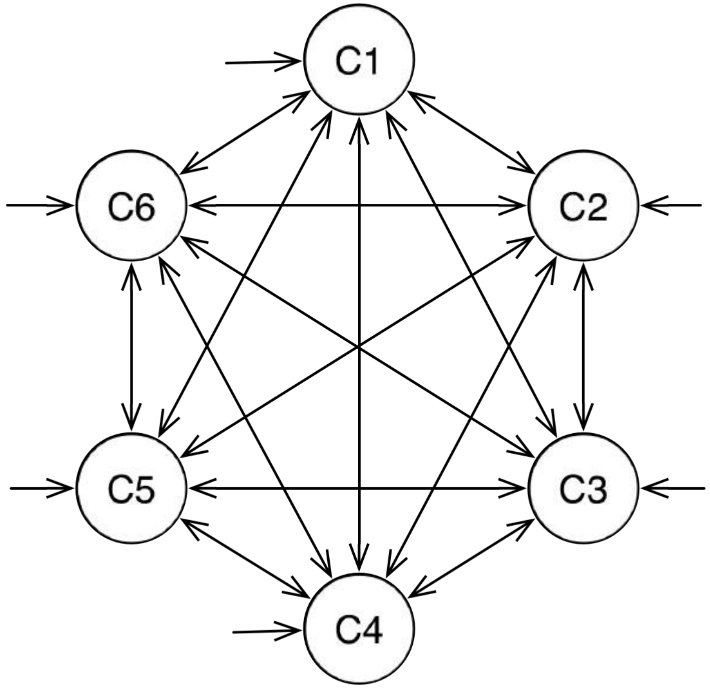
C1 = {A, E}
C2 = {A, E, F}
C3 = {A, B, C, E}
C4 = {A, B, C, E, F}
C5 = {A, B, D, E}
C6 = {A, B, D, E, F}



Temporal properties

- A temporal property is a condition over the executions of a transition system



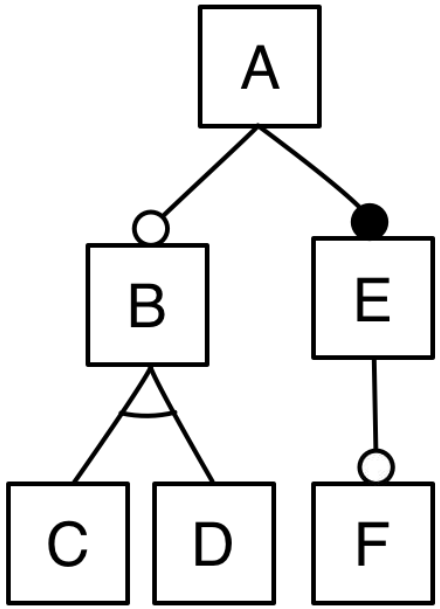


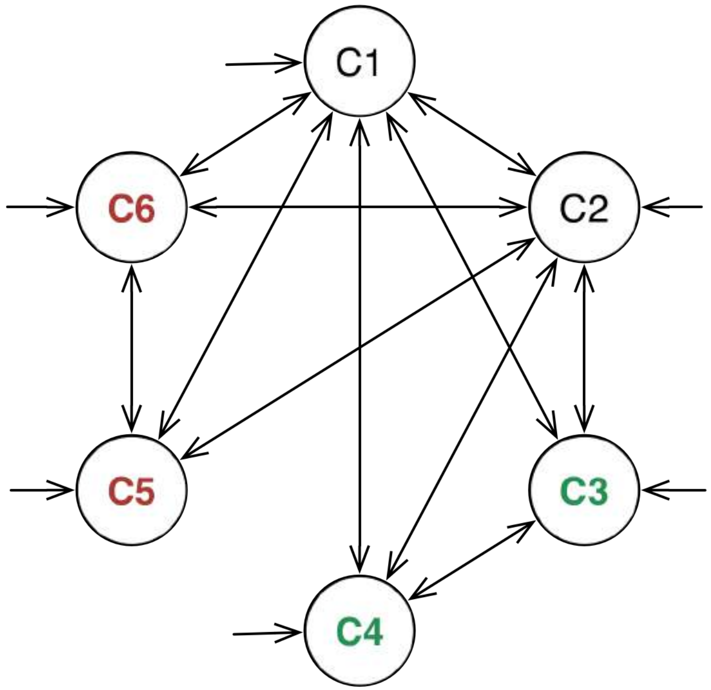
C1 = {A, E}
C2 = {A, E, F}
C3 = {A, B, C, E}
C4 = {A, B, C, E, F}
C5 = {A, B, D, E}
C6 = {A, B, D, E, F}

$$P = \{s_0 s_1 s_2 s_3 \ldots \mid C \in L(s_i) \leftrightarrow D \notin L(s_{i+1})\}$$

Temporal properties

- A temporal property is a condition over the executions of a transition system





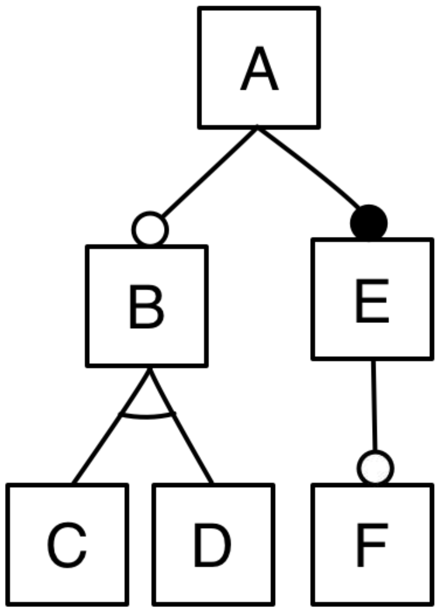
C1 = {A, E}
C2 = {A, E, F}
C3 = {A, B, **C**, E}
C4 = {A, B, **C**, E, F}
C5 = {A, B, **D**, E}
C6 = {A, B, **D**, E, F}

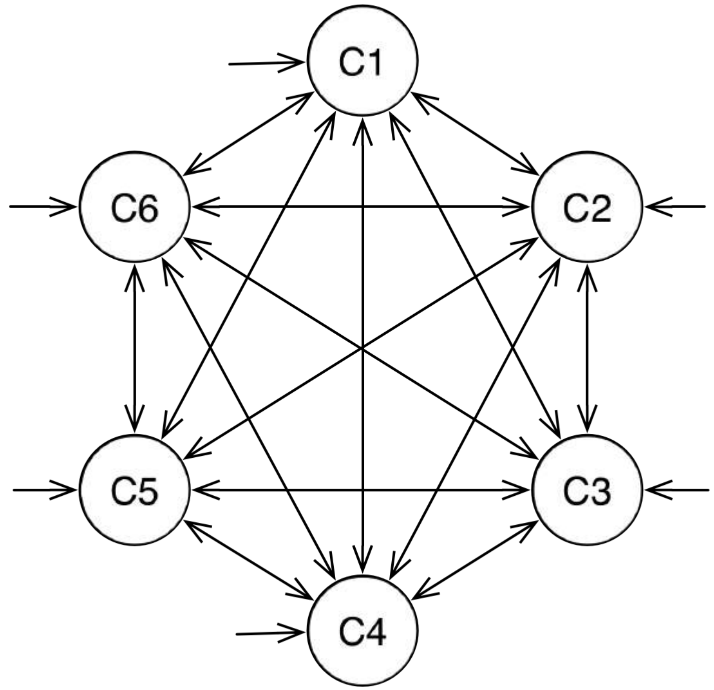
$$P = \{s_0 s_1 s_2 s_3 \ldots \mid C \in L(s_i) \leftrightarrow D \notin L(s_{i+1})\}$$

Linear Temporal Logic (LTL)

- Defines temporal properties over transition systems
- Combines propositional logic with temporal operators (always, eventually, until)
  - $\Box A$ // always A
  - $\Box(M2 \rightarrow \neg \bigcirc M1)$ // always (M2 is not followed by M1)
  - $\Box(M2 \rightarrow \neg \diamond M1)$ // after M2, M1 is not allowed

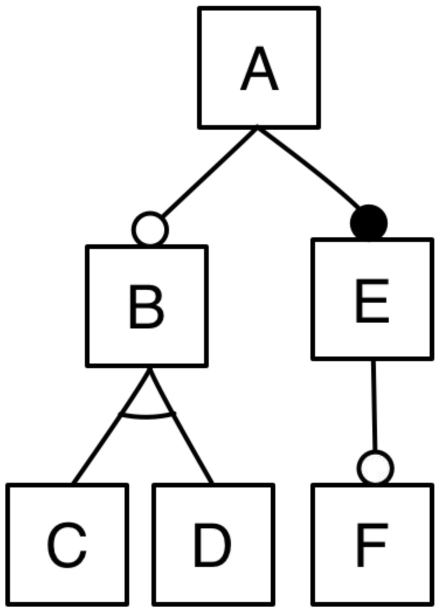
DSPL with temporal properties



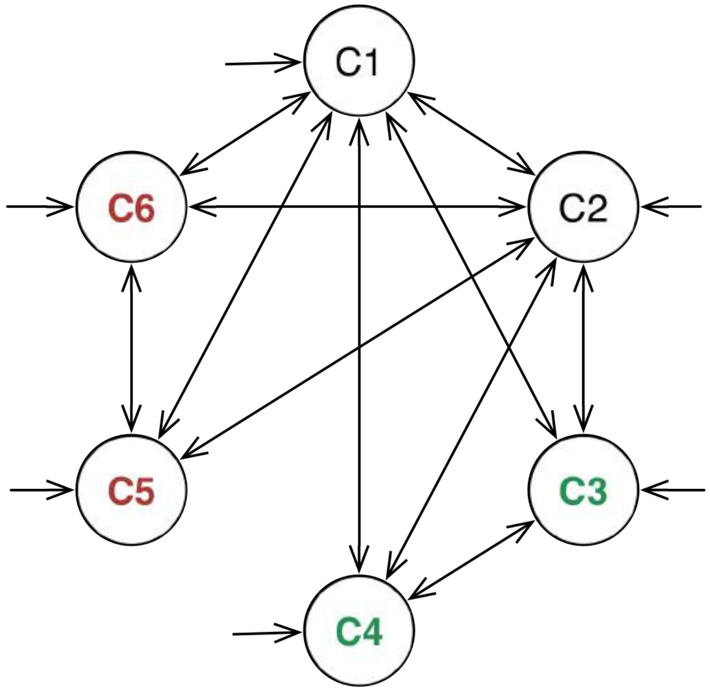


$$\square(C \rightarrow \neg \bigcirc D)$$
$$\square(D \rightarrow \neg \bigcirc C)$$

DSPL with temporal properties

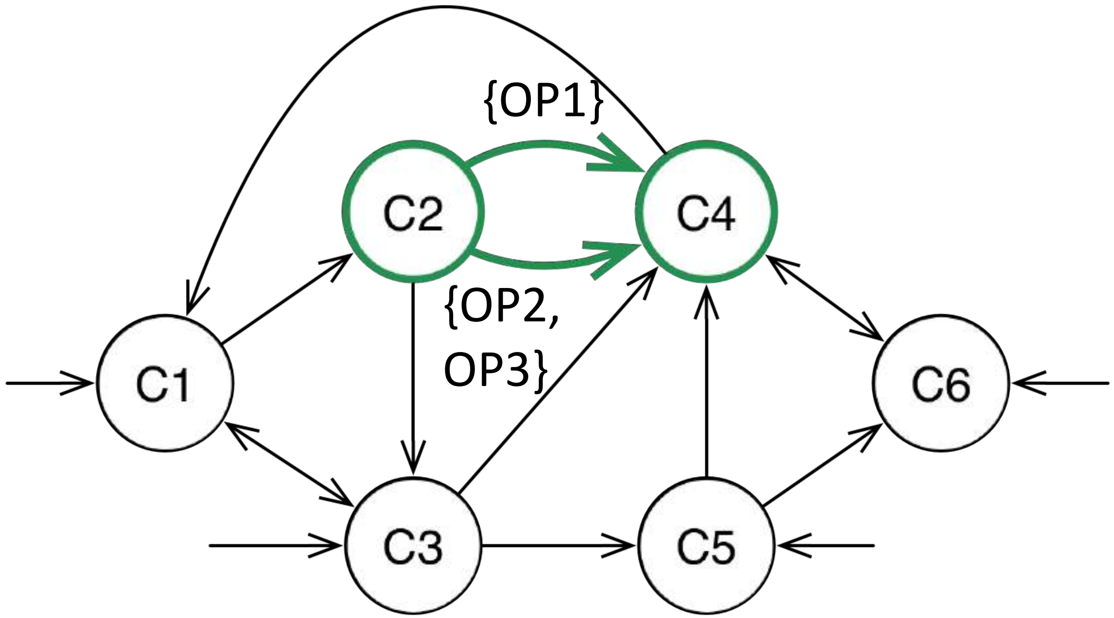


$$\Box(C \rightarrow \neg \bigcirc D)$$
$$\Box(D \rightarrow \neg \bigcirc C)$$



Reconfiguration operations

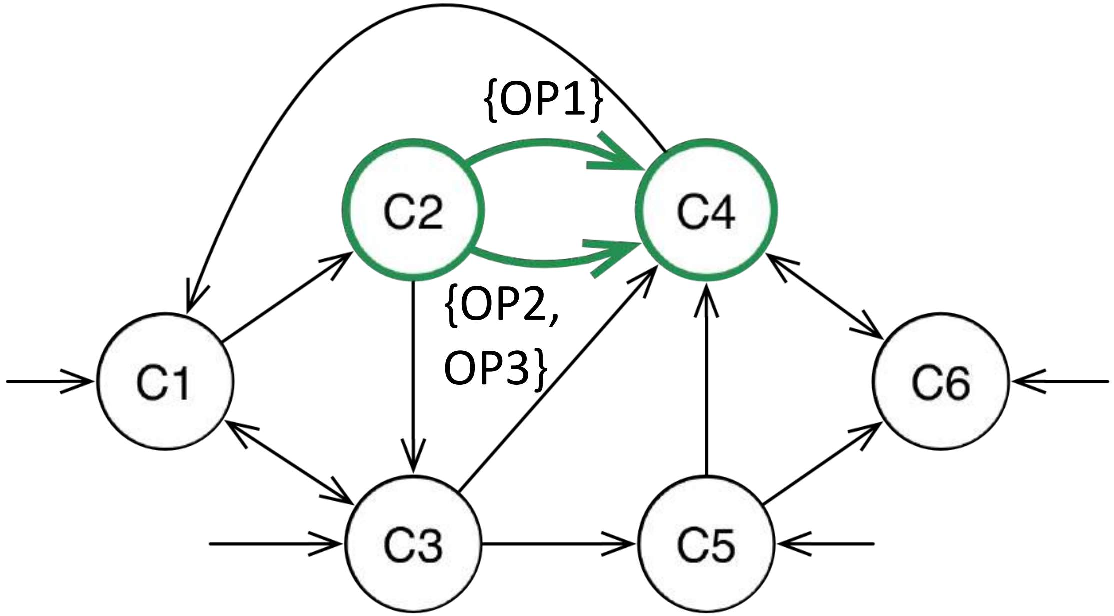
- Doubly labeled transition systems[22]



[22] M. H. ter Beek et al., "An Action/State-Based Model-Checking Approach for the Analysis of Communication Protocols for Service-Oriented Applications," in Proc. 12th Int. Workshop Formal Methods for Industrial Critical Systems (FMICS'07), Berlin, Germany, Jul. 2008, pp. 133–148.

Reconfiguration operations

- Doubly labeled transition systems[22]
  - $TS = (S, I, \boldsymbol{OP}, R, AP, L)$
  - OP is the set of reconfiguration operations in the DSPL
  - $R \subseteq S \times 2^{OP} \times S$



[22] M. H. ter Beek et al., "An Action/State-Based Model-Checking Approach for the Analysis of Communication Protocols for Service-Oriented Applications," in Proc. 12th Int. Workshop Formal Methods for Industrial Critical Systems (FMICS'07), Berlin, Germany, Jul. 2008, pp. 133–148.

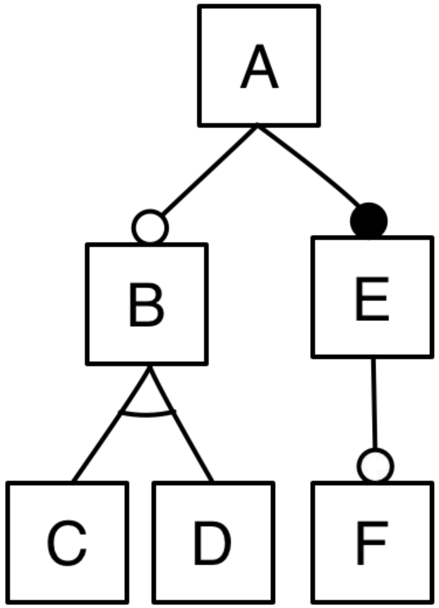
State/Event LTL

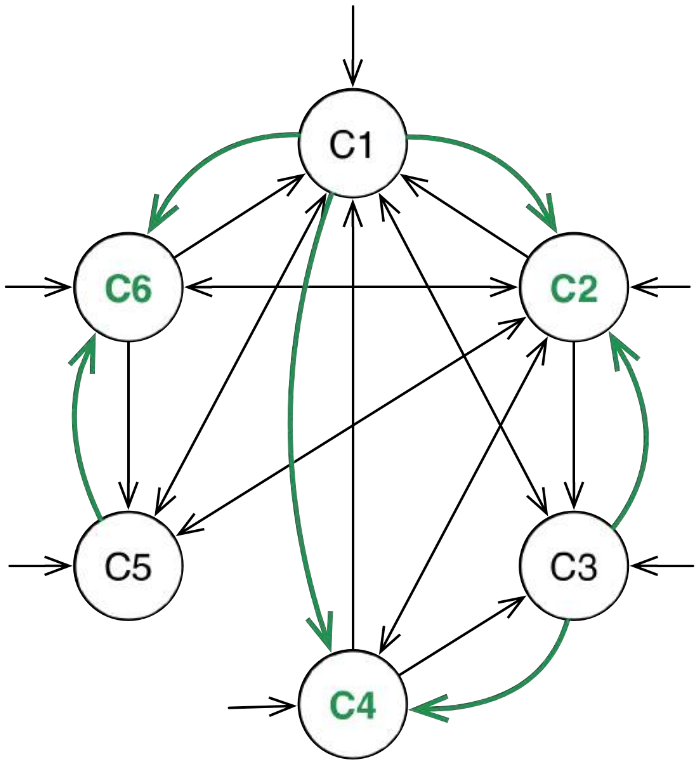
- SE-LTL can express temporal expressions over state and transition labels[23]
  - Can combine reconfiguration operations and features in temporal constraints

[23] S. Chaki et al. “State/Event-Based Software Model Checking,” in Proc. 4th Int. Conf. Integrated Formal Methods (IFM’04), Canterbury, UK, Apr. 2004, pp. 128–147.

Reconfiguration operations

$$OP = \{ActivateF\}$$





C1 = {A, E}
C2 = {A, E, F}
C3 = {A, B, C, E}
C4 = {A, B, C, E, F}
C5 = {A, B, D, E}
C6 = {A, B, D, E, F}

$$\Box(C \rightarrow \neg \bigcirc D)$$
$$\Box(D \rightarrow \neg \bigcirc C)$$

$$\Box((\neg F \wedge \bigcirc F) \leftrightarrow ActivateF)$$

Examples

- Cannot downgrade MySQL plan
  - [](M2 -> !<>M1)
  - [](M3 -> !<>(M1 | M2))
  - [](M4 -> !<>(M1 | M2 | M3))
  - [](Change(ClearDBMySQL) -> UpgradeClearDB)
- Upgrade PostgreSQL
  - [](Change(HerokuPostgres) & (H1 | H2) -> PGCopy)
  - [](Change(HerokuPostgres) -> (PGCopy | FollowerChangeover))
- Change Location
  - [](Change(Location) -> MigrateApp)

Problem statement

- How to model constraints over the adaptation behavior?
  – Temporal dependencies between features and reconfiguration operations

- How to reason over a variability model with reconfiguration constraints to find reconfigurations that meet a given criteria?
  – e.g. reduced downtime or costs

Reasoning

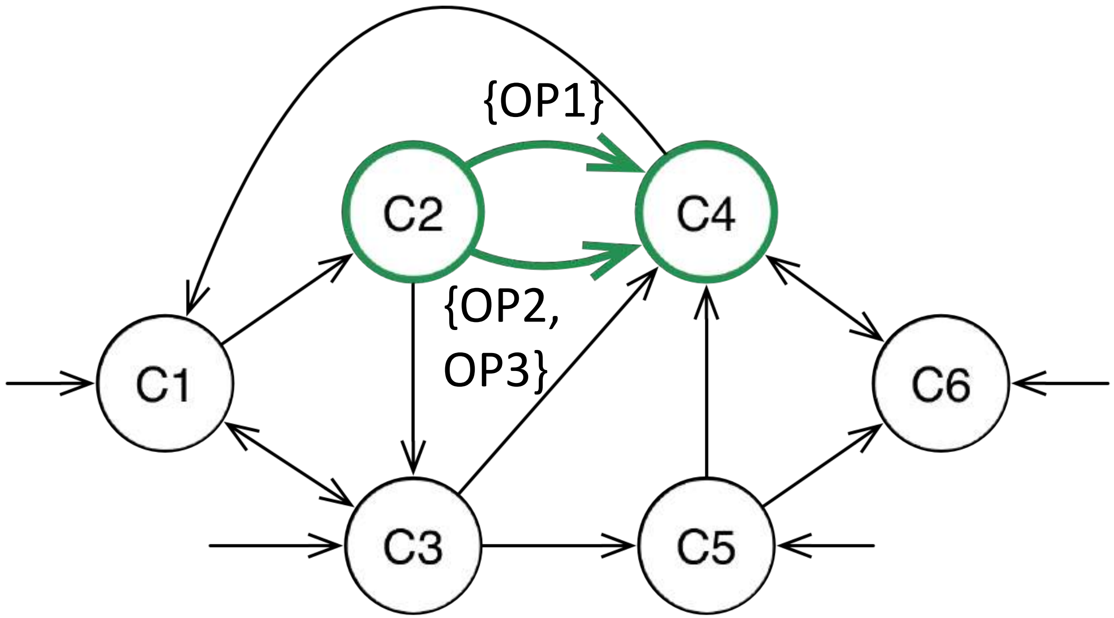
- Reconfiguration request
  - Features to be included/excluded

Reasoning

- Reconfiguration request
  - Features to be included/excluded
- Cost-based constraints
  - Reconfiguration time, downtime, financial cost, etc

Reasoning

- Reconfiguration query: $Q = (A, E, \phi)$
  - A: features to include
  - E: features to exclude
  - ϕ: constraint over costs
- Example query: $q = (\{C\}, \{D\}, \text{downtime} = 0)$



Symbolic Representation

- Building the transition system for a feature model can be unfeasible
  - State-explosion problem
- Represent a transition system as a propositional formula
  - Use SAT solver to solve reconfiguration queries

Symbolic Representation

- Feature models[25] and SE-LTL expressions[27] can be represented as propositional formulas

$$\widetilde{M} \wedge \widetilde{x} \wedge \widetilde{M}' \wedge \widetilde{s} \wedge \widetilde{q}$$

  - $\widetilde{M}$ and $\widetilde{M}'$ represent the set of possible source and target states (configurations of the feature model M)
  - $\widetilde{x}$ is the conjunction of LTL expressions
  - $\widetilde{s}$ represents the current state
  - $\widetilde{q}$ represents the reconfiguration query (pseudo-boolean encoding)

[25] D. Batory, "Feature Models, Grammars, and Propositional Formulas," in Proc. 9th Int. Conf. Software Product Lines (SPLC'05), Rennes, France, Sep. 2005, pp. 7–20.

[27] A. Cimatti, M. Pistore, M. Roveri, and R. Sebastiani, "Improving the Encoding of LTL Model Checking into SAT," in Proc. 3rd Int. Workshop Model Checking and Abstract Interpretation (VMCAI'02), Venice, Italy, Jan. 2002, pp. 196–207.

Problem statement

- How to model constraints over the adaptation behavior?
  - Temporal dependencies between features and reconfiguration operations
- How to reason over a variability model with reconfiguration constraints to find reconfigurations that meet a given criteria?
  - e.g. reduced downtime or costs

Evaluation

Evaluation

- Case study on Heroku PaaS
  - feasibility for modeling reconfiguration constraints
  - performance of reasoning

Evaluation

- Case study on Heroku PaaS
- Feature Model extracted from documentation
  - 7 available regions, 11 programming frameworks, 6 container sizes
  - reconfiguration constraints
  - 161 addon services (data storage, networking, security, …)
  - **1036 features, 134 cross-tree constraints, 124 temporal constraints**

Evaluation

- Case study on Heroku PaaS
- Feature Model extracted from documentation
- Simulate context changes
  - 4 adaptation scenarios
  - 5 reconfiguration queries
  - 3 utilization profiles
  - 12 executions

Evaluation

- Case study on Heroku PaaS
- Feature Model extracted from documentation
- Simulate context changes
- Adaptation scenarios
  - Change in database size requires new database plan
  - Request for a new feature not available in current region
  - Change in programming framework and database
  - Scaling up application container

Evaluation

- Case study on Heroku PaaS
- Feature Model extracted from documentation
- Simulate context changes
- Adaptation scenarios
- Reconfiguration queries
  - No constraints
  - Constraints over price
  - Constraints over downtime/price
  - Optimize on price
  - Optimize on downtime/price

Evaluation

- Case study on Heroku PaaS
- Feature Model extracted from documentation
- Simulate context changes
- Adaptation scenarios
- Reconfiguration queries
- Application utilization profiles
  - Database size, application size, startup time, etc…

Evaluation

- Case study on Heroku PaaS
- Feature Model extracted from documentation
- Simulate context changes
- Adaptation scenarios
- Reconfiguration queries
- Application utilization profiles
  - DBSize: 10GB, AppSize: 100 MB, AppStartUp: 15
  - DBSize: 100GB, AppSize: 200 MB, AppStartUp: 30s
  - DBSize: 2TB, AppSize: 500 MB, AppStartUp: 60s

Results

Process step	Execution time (ms) Avg	StdDev	Min	Max	#Exec
Build Trans System	8777.31	303.71	8262	10308	720
- Process FM	244.75	28.57	191	552	
- Process LTL	8533.57	291.81	8025	10023	
All Queries	183.34	50.20	118	389	720
- Build	83.05	50.69	27	227	
- Solve	100.29	37.13	5	200	
Q wo/ Constraints	140.97	12.93	118	198	144
- Build	32.73	4.56	27	48	
- Solve	108.24	11.58	90	153	
Q w/ Constraints	224.23	55.55	128	389	288
- Build	140.41	27.65	80	227	
- Solve	83.82	53.13	5	200	
Q w/ Optimization	163.63	13.26	136	230	288
- Build	50.85	7.11	38	77	
- Solve	112.77	10.21	94	166	

Results

	Execution time (ms)				
Process step	**Avg**	**StdDev**	**Min**	**Max**	**#Exec**
Build Trans System	8777.31	303.71	8262	10308	720
- Process FM	244.75	28.57	191	552	
- Process LTL	8533.57	291.81	8025	10023	
All Queries	183.34	50.20	118	389	720
- Build	83.05	50.69	27	227	
- Solve	100.29	37.13	5	200	
Q wo/ Constraints	140.97	12.93	118	198	144
- Build	32.73	4.56	27	48	
- Solve	108.24	11.58	90	153	
Q w/ Constraints	224.23	55.55	128	389	288
- Build	140.41	27.65	80	227	
- Solve	83.82	53.13	5	200	
Q w/ Optimization	163.63	13.26	136	230	288
- Build	50.85	7.11	38	77	
- Solve	112.77	10.21	94	166	

Results

	Execution time (ms)				
Process step	**Avg**	**StdDev**	**Min**	**Max**	**#Exec**
Build Trans System	8777.31	303.71	8262	10308	720
- Process FM	244.75	28.57	191	552	
- Process LTL	8533.57	291.81	8025	10023	
All Queries	183.34	50.20	118	389	720
- Build	83.05	50.69	27	227	
- Solve	100.29	37.13	5	200	
Q wo/ Constraints	140.97	12.93	118	198	144
- Build	32.73	4.56	27	48	
- Solve	108.24	11.58	90	153	
Q w/ Constraints	224.23	55.55	128	389	288
- Build	140.41	27.65	80	227	
- Solve	83.82	53.13	5	200	
Q w/ Optimization	163.63	13.26	136	230	288
- Build	50.85	7.11	38	77	
- Solve	112.77	10.21	94	166	

Results

- Temporal constraints enhance modeling of DSPLs
  - Compact notation for constraints over transitions
  - Support for reasoning over reconfiguration operations

- Performance is acceptable in the cloud context
  - Implementation can be improved

- Threats to validity
  - Case study is not exhaustive and considers only cloud computing

Conclusion & Perspectives

- Temporal constraints in DSPL
  - Better modeling of adaptive behavior
  - Reasoning over adaptation alternatives

Conclusion & Perspectives

- Temporal constraints in DSPL
  - Better modeling of adaptive behavior
  - Reasoning over adaptation alternatives
- Cardinality-based feature models
- Multi-cloud environment adaptation

Questions

Gustavo Sousa

gustavo.sousa@inria.fr

More information

http://researchers.lille.inria.fr/sousa/seams2017/