CD2Alloy: A Translation of Class Diagrams to Alloy

Shahar Maoz and Jan Oliver Ringert and Bernhard Rumpe
The publications of the Department of Computer Science of *RWTH Aachen University* are in general accessible through the World Wide Web.

http://aib.informatik.rwth-aachen.de/
CD2Alloy: A Translation of Class Diagrams to Alloy

Shahar Maoz
Jan Oliver Ringert\textsuperscript{1}
Bernhard Rumpe

\textsuperscript{1}J.O. Ringert is supported by the DFG GK/1298 AlgoSyn.
Abstract

This document defines a translation from UML/P class diagrams [Rum04, Sch12] to Alloy modules [Jac06] and a translation from corresponding Alloy instances back to UML/P object diagrams. An overview of the translation was first presented in [MRR11]. We define the translations and give an example, showing a class diagram and its complete representation in Alloy, as well as a corresponding Alloy instance and its complete representation back in an object diagram.
Chapter 1

Class Diagrams

1.1 Grammar of UML/P CDs

We show an abbreviated version of the grammar of the class diagrams language used as input for the CD2Alloy translation. Relevant parts of the grammar of UML/P CDs [Rum04, Sch12] are shown in Listing 1.1. The production rules to express cardinalities of association ends and modifiers of CD elements are part of the Common grammar (parent of CD grammar) shown in Listing 1.2. The grammars in the listings are adapted from [Sch12].

```plaintext
grammar CD extends mc.uml.p.common.Common {
  abstract CDElement;
  abstract CDType extends CDElement;
  external Value, Body;

  CDDefinition =
    Completeness? Stereotype? "classdiagram" Name
    "{" ( CDClass | CDInterface | CDEnum | CDAssociation
    | Invariant )* "}";

  CDClass astextends CDType =
    Completeness? Modifier? "class" Name TypeParameters?
    ( "extends" superclasses:ReferenceType
    ("," superclasses:ReferenceType)* )?
    ( "implements" interfaces:ReferenceType
    ("," interfaces:ReferenceType)* )?
    ( ";" | "{" ( CDAttribute | CDConstructor | CDMethod)*
    "}" );

  CDInterface astextends CDType =
}
Completeness? Modifier? "interface" Name
  TypeParameters?
  ( "extends" interfaces:ReferenceType
    ("," interfaces:ReferenceType)* )?
  ( ";" | "{" (CDAttribute | CDMethod)* "}" );

dEnum astextends CDType =
  Completeness? Modifier? "enum" Name
  ( "implements" interfaces:ReferenceType
    ("," interfaces:ReferenceType)* )?
  ( ";" | "{" (CDEnumConstant ("," CDEnumConstant)* "");
    (CDAttribute | CDConstructor | CDMethod)* )? "}" );

CDEnumConstant = Name ( "(" CDEnumParameter
  ("," CDEnumParameter)* ")" )?

CDEnumParameter = Value;

CDAttribute = Modifier? Type Name ("=" Value)? ";";

CDQualifier = Name | Type;

CDAssociation astextends CDElement =
  Stereotype? ( ["association"] | ["aggregation"] | ["composition"] )
  Derived:["/"]? Name?
  leftModifier:Modifier? leftCardinality:Cardinality?
  leftReferenceName:QualifiedName
  ( ["" leftQualifier:CDQualifier "]")?
  ("(" leftRole:Name ")")?
  ( leftToRight:["- >"] | rightToLeft:["< -"] | bidirectional:["< - >"] | simple:["--"] )
  ("(" rightRole:Name ")")?
  ("[" rightQualifier:CDQualifier "]")?
  rightReferenceName:QualifiedName
  rightCardinality:Cardinality?
  rightModifier:Modifier? ";";

Listing 1.1: Extract of the UML/P class diagram grammar given in MontiCore format

grammar Common extends mc.types.Types {
Cardinality =
  "[" ( many:["*"] | lowerBoundLit:IntLiteral ( ".."
    ( upperBoundLit:IntLiteral | noUpperLimit:["*"]) )?
  "])";
1.2 Example CD

Consider the example CD shown in Fig. 1.1 and given in textual UML/P syntax in Listing 1.3. The class diagram contains 4 enumerations, 9 classes, and 6 associations.

An employee works in at least one up to three addresses, owns one or more insurances, and can be assigned to an unlimited number of companies. Insurances have a kind attribute with the possible enumeration values workAccident, transport, and international. Companies can have an unlimited number of cars. Drivers are special employees that have a driving experience with the possible enumeration values beginner or expert. Drivers can optionally have a license and drive up to one vehicle. Vehicles have to be driven by exactly one driver. The class Vehicle is an abstract class and there are two concrete types of vehicles: Car and Truck. Cars have a color attribute with possible enumeration values black, red, and white and a kind attribute with possible enumeration values sportsCar, van, limousine, and regular.
Figure 1.1: $cd_1$, an example class diagram with classes, attributes, enumerations, associations with multiplicities, and inheritance.
package examples.cD2AlloyJournalPaperExamples;

classdiagram CD2AlloyExample {

    enum ColorKind {black, red, white;}
    enum CarKind {sportsCar, van, limousine, regular;}
    enum InsuranceKind {workAccident, transportation, international;}
    enum DrivingExperience {expert, beginner;}

class Employee;
class Driver extends Employee {
    DrivingExperience drivingExperience;
}
class Address;
abstract class Vehicle {
    Date registrationDate;
    String licensePlate;
}
class Car extends Vehicle {
    ColorKind color;
    CarKind kind;
}
class Truck extends Vehicle;
class Company {}
class License {}
class Insurance {
    InsuranceKind kind;
}

association Driving [1] Driver (drivenBy) --
    (drives) Car [0..1];
association WorkPlace [1] Employee (of) ->
    (worksIn) Address [1..3];
association CompaniesCars [0..1] Company (of) ->
    (cars) Car [*];
association CompaniesEmps [*] Company (of) ->
    (emps) Employee [*];
composition EmployeeInsurances [1] Employee (of) ->
    (insurances) Insurance [1..*];
association DriversLicense [1] Driver (ownedBy) --
    (license) License [0..1];
}

Listing 1.3: The class diagram cd\textsubscript{1} (CD2AlloyExample) given in UML/P CD concrete syntax
Chapter 2

The CD2Alloy Translation

We present our complete translation of CDs given in abstract syntax to the Alloy language in Listing 2.1 (foundational part) and Figures 2.1-2.3 (parts dependent on CD).

2.1 The foundational part

Every translated class diagram in Alloy consists of a constant foundational part that is shown in Listing 2.1. This part is generic, that is, common to all generated modules, independent of the input CD at hand.

Listing 2.1 shows the abstract signature FName (lines 9-11) used to represent association role names and attribute names for all classes in the module. The abstract signature Obj (lines 22-24) is the parent of all classes in the module; its get Alloy field relates it and an FName to instances of Obj, Val, and EnumVal. The abstract signature Val (line 27), which we use to represent all predefined types (i.e., primitive types and other types that are not defined as classes in the CD). Values of enumeration types are represented using the signature EnumVal (line 14). Enumeration values and primitive values should only appear in an instance if they are referenced by an object (as specified by the facts in lines 16-19 and lines 29-32).

Listing 2.1 shows some of the generic, parametrized predicates responsible for specifying the relation between objects and fields: ObjAttrib (lines 38-41) limits objs.get[fName] to the correct field’s type and ensures that there is exactly one object, value, or enumeration value related to the field name by the get relation; ObjFNames (lines 34-36) is used to ensure objects do not have field names other than the ones stated in the CD. Listing 2.1 also shows one of the generic predicates responsible for specifying association multiplicities: ObjUAttrib (lines 48-51) provides an upper bound for the number of objects
in the set represented by the `get` relation for a specified role name.

```plaintext
1 // //////////////////////////////////////////////////
2 // Generic Head of CD Model - Oct. 11, 2012
3 // //////////////////////////////////////////////////
4
5 // Some markers not belonging to the model
6 one sig auxiliary {}
7
8 // Names of fields/associations in classes of the model
9 abstract sig FName {
10   is: one auxiliary
11 }
12
13 // Names of enum values in enums of the model
14 abstract sig EnumVal {}
15
16 fact enums {
17   // no enum values can exist on their own
18   all v: EnumVal | some f: FName | v in Obj.get[f]
19 }
20
21 // Parent of all classes relating fields and values
22 abstract sig Obj {
23   get : FName -> { Obj + Val + EnumVal }
24 }
25
26 // Values of fields
27 abstract sig Val {}
28
29 fact values {
30   // no values can exist on their own
31   all v: Val | some f: FName | v in Obj.get[f]
32 }
33
34 pred ObjFNames[objs : set Obj, fName : set FName] {
35   no objs.get[FName - fName]
36 }
37
38 pred ObjAttrib[objs : set Obj, fName : one FName, fType : set {
39   Obj + Val + EnumVal }] {
40   objs.get[fName] in fType
41   all o: objs | one o.get[fName]
42 }
43
44 pred ObjMeth[objs : set Obj, fName : one FName, fType : set {
45   Obj + Val + EnumVal }] {
46   objs.get[fName] in fType
47   all o: objs | one o.get[fName]
48 }
```
pred ObjLUAttrib[objs : set Obj, fName: one FName, fType: set Obj, low: Int, up: Int] {
  ObjLattrib[objs, fName, fType, low]
  ObjUAttrib[objs, fName, fType, up]
}

pred ObjLattrib[objs : set Obj, fName: one FName, fType: set Obj, low: Int] {
  objs.get[fName] in fType
  all o: objs | (#o.get[fName] >= low)
}

pred ObjUAttrib[objs : set Obj, fName: one FName, fType: set Obj, up: Int] {
  objs.get[fName] in fType
  all o: objs | (#o.get[fName] <= up)
}

pred ObjLU[objs : set Obj, fName: one FName, fType: set Obj, low: Int, up: Int] {
  ObjL[objs, fName, fType, low]
  ObjU[objs, fName, fType, up]
}

pred ObjL[objs : set Obj, fName: one FName, fType: set Obj, low: Int] {
  all r: objs | # { l: fType | r in l.get[fName]} >= low
}

pred ObjU[objs : set Obj, fName: one FName, fType: set Obj, up: Int] {
  all r: objs | # { l: fType | r in l.get[fName]} <= up
}

pred BidiAssoc[left : set Obj, lFName: one FName, right : set Obj, rFName: one FName] {
  all l: left | all r: l.get[lFName] | l in r.get[rFName]
  all r: right | all l: r.get[rFName] | r in l.get[lFName]
}

pred Composition[left: set Obj, lFName: some FName, right: set Obj] {
  // all l1, l2: left | (# {l1.get[lFName] & l2.get[lFName]} > 0) => l1=l2
  all r: right | # { l: left, lF: lFName | r in l.get[lF]} = 1
}
Listing 2.1: The foundational part of the CD2Alloy translation with basic signatures `Obj`, `EnumVal`, `Val`, `FName`, and predicates to describe class diagrams' semantics inside Alloy

### 2.2 CD specific signatures and functions

Rules to create parts of the generated Alloy module depending on the class diagram are shown in Figures 2.1, 2.2, and 2.3. Elements of the abstract syntax of the class diagram are accessed using the names of the production rules from the grammar in Listing 1.1, e.g., `CDClass` and `CDAssociation` as abstract syntax nodes of classes and associations. The attribute names from the grammar, e.g., `superclasses` and `name`, are used for navigation in the abstract syntax.

The input of the translation rules is the abstract syntax of a class diagram. The output produced by the rules is given in Alloy syntax.

The first part of rule 1 from Fig. 2.1 creates a signature for every class in the class diagram extending the common signature `Obj`. The instances of these signatures represent objects in an object model in the semantics of the CD. The second part of rule 1 creates functions that return all atoms of all sub classes for each class in the CD. These functions allow to express sub classing.

Rule 2 introduces a function for every interface in the CD that returns all instances of classes implementing the interface.

Rule 3 introduces an Alloy singleton signature for every field name of an attribute of a class in the CD (part 1) and for every role name of associations (part 2). All these signatures extend the abstract signature `fName` and can be used in the `Obj.get` relation.

Rule 4 creates for every attribute’s type that is not a class in the CD a signature extending signature `Val` whose at most one atom represents an instance of a primitive or no further defined type.

The first part of rule 5 creates a signature for every enumeration value in the CD. The second part of rule 5 creates a function for every enumeration type.

The first part of rule 6 creates a function for every “part” in a composite whole-part-relation in the CD that returns all “wholes” in the CD that might have this “part” as a part. The second part of rule 6 creates a similar function that returns all field names that the “wholes” use to navigate to their “parts”.
2.3 The predicate \textit{pred cd}

The second set of rules in Fig. 2.2 and Fig. 2.3 describes how the Alloy predicate \textit{pred cd} is generated. This predicate describes the semantics of the CD based on the generic part and the generated signatures and functions. The first set of rules in Fig. 2.2 handles classes and the \texttt{Obj.get} relation. The second set of rules A1-A3 in Fig. 2.3 describe the generation of predicates for associations.

The first rule in Fig. 2.2 uses predicate \texttt{ObjAttrib} to declare the attributes for every class in the class diagram. Note, that the list of attributes includes also all inherited attributes of the class since there is no inheritance on the Alloy signature level for the classes of the CD.

Rule 2 restricts the elements of the classes’ \texttt{get} relation to their attributes and role names of their partners in (inherited) associations.

Rule 3 specifies that all signatures representing abstract classes will have no atoms in an Alloy instance and that signatures representing classes marked as singleton will contain exactly one atom.

Rule 4 generates an instantiation of the predicate \textit{Composition} for every class which is the in the part-relation of a composition. This predicate forces Alloy instances to only associate the atoms of the part representing signature to exactly one atom of of the possible wholes via the \texttt{get} relation.

Rule 5 restricts all objects in object models of the CD to be instances of the classes of the CD. This constraint is important for multiple CD analysis where the Alloy module might contain signatures extending \texttt{Obj} that represent classes of other class diagrams.

Rule A1 in Fig. 2.3 specifies a restriction on the sets of links (entries in the \texttt{Obj.get} relation) off bidirectional associations using the predicate \texttt{BidiAssoc} from Listing 2.1.

Rule A2 applies to bidirectional associations (syntax <-> and --) and those defined from right to left (syntax <-). The multiplicities and target objects of links are defined using predicates for lower and upper bounds (e.g., \texttt{ObjLU}) or lower bounds only (e.g., \texttt{ObjL}). Multiplicities of a single constant $k$ are internally expressed as the range $k..k$ and the range $k..*$ and multiplicity $*$ both as the lower bounds $k$ and $0$. For the side navigable in the CD by the partner’s role name predicates with suffix \texttt{Attrib} are used.

Rule A3 is the analog to rule A2 and applies to bidirectional associations (syntax <-> and --) and those defined from left to right (syntax ->).
∀c ∈ CDClass:
    sig c.name extends Obj {}
∀c ∈ CDClass:
    fun c.nameSubsCD {
        ∀cSub ∈ CDClass s.t. c ∈ cSub.superclasses⁺:
            ∑ cSub.name}
2 ∀i ∈ CDInterface:
    fun i.nameSubsCD {
        ∀cImpl ∈ CDClass s.t. i ∈ cImpl.interfaces⁺:
            ∑ cImpl.name}
3 ∀n ∈ {a.name | c ∈ CDClass, a ∈ c.cdAttribute}:
    one sig n extends fName {}
∀rn ∈ {a.leftRole, a.rightRole | a ∈ CDAssociation}:
    one sig rn extends fName {}
4 ∀t ∈ {a.type | c ∈ CDClass, a ∈ c.cdAttribute s.t. a.type /∉ CDClasses}:
    lone sig type_t extends Val {}
5 ∀e ∈ CDEnum:
    ∀enumVal ∈ e.cdEnumConstant:
        lone sig enum_e.name_enumVal.name extends EnumVal {}
∀e ∈ CDEnum:
    fun e.nameEnumCD: set EnumVal {
        ∀enumVal ∈ e.cdEnumConstant:
            ∑ enumVal.name }
6 ∀part ∈ {comp.rightReferenceName |
            comp ∈ CDAssociation ∧ comp.isComposition}:
    fun partCompositesCD: set Obj {
        ∀comp’ ∈ CDAssociation s.t. comp’.isComposition ∧
            comp’.rightReferenceName = part:
                ∑ comp’.leftReferenceNameSubsCD }
fun partCompFieldNamesCD: set FName {
    ∀comp’ ∈ CDAssociation s.t. comp’.isComposition ∧
        comp’.rightReferenceName = part:
            ∑ comp’.rightRole }

Figure 2.1: Rules to generate signatures and functions based on a given class diagram.
pred cd {
1 \forall c \in CDClass:
    \forall a \in (CDClass.superclasses^*).cdAttribute:
    ObjAttrib[c.name, a.name, a.type]
2 \forall c \in CDClass:
    ObjFNames[c.name, \forall a \in (CDClass.superclasses^*).cdAttribute:
    \sum a.name
    \forall rname \in \{a.leftRole \mid a \in CDAssociation \land
    a.rightReferenceName \in \{sc.name \mid sc \in c.superclasses^*\}\} \cup
    \{a.rightRole \mid a \in CDAssociation \land
    a.leftReferenceName \in \{sc.name \mid sc \in c.superclasses^*\}\}
    + \sum rname ]
3 \forall c \in CDClass s.t. c.isAbstract:
    no c.name
\forall c \in CDClass s.t. c.isSingleton:
    one c.name
4 \forall rname \in \{comp.rightReferenceName \mid
    comp \in CDAssociation \land comp.isComposition\}:
    Composition[rnameCompositesCD, rnameCompFieldNamesCD, rname]
5 \forall c \in CDClass:
    Obj = \sum c.name
}
Figure 2.2: The first set of rules to generate the predicate that describes the semantics of the CD based on the generic predicates from Listing 2.1 and the signatures and functions generated from rules 1-5 from Fig. 2.1.
∀assoc ∈ CDAssociation :

A1 if assoc.isBidirectional then
    BidiAssoc[assoc.leftReferenceName, assoc.rightRole,
               assoc.rightReferenceName, assoc.leftRole]

A2 if assoc.isBidirectional ∨ assoc.isRightToLeft then
    if assoc.leftCardinality.isLowerUpper then
        ObjLUAttrib[ assoc.rightReferenceName, assoc.leftRole,
                      assoc.leftReferenceName, assoc.leftCardinality.lower
                      assoc.leftCardinality.upper ]
    else
        ObjLAttrib[ assoc.rightReferenceName, assoc.leftRole,
                      assoc.leftReferenceName, assoc.leftCardinality.lower ]
    if not assoc.isBidirectional then
        if assoc.rightCardinality.isLowerUpper then
            ObjLU[ assoc.leftReferenceName, assoc.leftRole,
                   assoc.rightReferenceName, assoc.rightCardinality.lower
                   assoc.rightCardinality.upper ]
        else
            ObjL[ assoc.leftReferenceName, assoc.leftRole,
                  assoc.rightReferenceName, assoc.rightCardinality.lower ]

A3 if assoc.isBidirectional ∨ assoc.isLeftToRight then
    if assoc.rightCardinality.isLowerUpper then
        ObjLUAttrib[ assoc.leftReferenceName, assoc.rightRole,
                     assoc.rightReferenceName, assoc.rightCardinality.lower
                     assoc.rightCardinality.upper ]
    else
        ObjLAttrib[ assoc.leftReferenceName, assoc.rightRole,
                     assoc.rightReferenceName, assoc.rightCardinality.lower ]
    if not assoc.isBidirectional then
        if assoc.leftCardinality.isLowerUpper then
            ObjLU[ assoc.rightReferenceName, assoc.rightRole,
                   assoc.leftReferenceName, assoc.leftCardinality.lower
                   assoc.leftCardinality.upper ]
        else
            ObjL[ assoc.rightReferenceName, assoc.rightRole,
                  assoc.leftReferenceName, assoc.leftCardinality.lower ]

Figure 2.3: The rules about associations to generate the predicate that describes the semantics of the CD based on the generic predicates from Listing 2.1 and the signatures and functions generated from rules 1-5 from Fig. 2.1.
Chapter 3

Example of Generated Alloy Module

The Alloy code shown in Listing 3.1 is generated from class diagram $cd_1$ presented in Sect. 1.2. The listing contains the CD specific part generated using the rules from Sect. 2.2 to generate CD specific signatures and functions (lines 1-109) and the rules from Sect. 2.3 to generate the predicate $\text{pred } cd$ (lines 113-166).

```alloy
// Concrete names of fields in cd
one sig drivenBy extends FName {}
one sig insurances extends FName {}
one sig licensePlate extends FName {}
one sig kind extends FName {}
one sig emps extends FName {}
one sig of extends FName {}
one sig drivingExperience extends FName {}
one sig registrationDate extends FName {}
one sig color extends FName {}
one sig worksIn extends FName {}
one sig cars extends FName {}
one sig ownedBy extends FName {}
one sig drives extends FName {}
one sig license extends FName {}

// Concrete value types in model cd
lone sig type_String extends Val {}
lone sig type_Date extends Val {}

// Concrete enum values in model cd
lone sig enum_CarKind_sportsCar extends EnumVal {}
lone sig enum_ColorKind_black extends EnumVal {}
lone sig enum_CarKind_van extends EnumVal {}
```
lone sig enum_ColorKind_red extends EnumVal {}

lone sig enum_InsuranceKind_transportation extends EnumVal {}

lone sig enum_CarKind_limousine extends EnumVal {}

lone sig enum_CarKind_regular extends EnumVal {}

lone sig enum_ColorKind_white extends EnumVal {}

lone sig enum_DrivingExperience_beginner extends EnumVal {}

lone sig enum_DrivingExperience_expert extends EnumVal {}

lone sig enum_InsuranceKind_international extends EnumVal {}

lone sig enum_InsuranceKind_workAccident extends EnumVal {}

// Classes in model cd

sig Driver extends Obj {}

sig Car extends Obj {}

sig License extends Obj {}

sig Truck extends Obj {}

sig Address extends Obj {}

sig Company extends Obj {}

sig Employee extends Obj {}

sig Insurance extends Obj {}

sig Vehicle extends Obj {}

// Interfaces in model cd

// Types wrapping subtypes

fun DriverSubsCD: set Obj {
  Driver
}

fun CarSubsCD: set Obj {
  Car
}

fun LicenseSubsCD: set Obj {
  License
}

fun TruckSubsCD: set Obj {
  Truck
}

fun AddressSubsCD: set Obj {
  Address
}

fun CompanySubsCD: set Obj {
  Company
}

fun InsuranceSubsCD: set Obj {

Insurance

fun EmployeeSubsCD: set Obj {
    Driver + Employee
}

fun VehicleSubsCD: set Obj {
    Car + Truck + Vehicle
}

// Types containing subtypes for definition of associations

// Types wrapping composite structures
fun InsuranceCompositesCD: set Obj {
    EmployeeSubsCD
}

// Types wrapping composite structures field names
fun InsuranceCompFieldNamesCD: set FName {
    insurances
}

// Enums
// Enum values in cd
fun InsuranceKindEnumCD: set EnumVal {
    enum_InsuranceKind_transportation +
    enum_InsuranceKind_workAccident +
    enum_InsuranceKind_international
}

fun ColorKindEnumCD: set EnumVal {
    enum_ColorKind_black + enum_ColorKind_red +
    enum_ColorKind_white
}

fun CarKindEnumCD: set EnumVal {
    enum_CarKind_sportsCar + enum_CarKind_van +
    enum_CarKind_limousine + enum_CarKind_regular
}

fun DrivingExperienceEnumCD: set EnumVal {
    enum_DrivingExperience_beginner +
    enum_DrivingExperience_expert
}

// Values and relations in cd
pred cd {
    // Definition of class Driver
// Definition of class Car
ObjAttrib[Car, licensePlate , type_String]
ObjAttrib[Car, kind , CarKindEnumCD]
ObjAttrib[Car, registrationDate , type_Date]
ObjAttrib[Car, color , ColorKindEnumCD]
ObjFNames[Car, licensePlate + kind + registrationDate +
        color + drivenBy + none]

// Definition of class License
ObjFNames[License, ownedBy + none]

// Definition of class Truck
ObjAttrib[Truck, licensePlate , type_String]
ObjAttrib[Truck, registrationDate , type_Date]
ObjFNames[Truck, licensePlate + registrationDate +
        drivenBy + none]

// Definition of class Address
ObjFNames[Address, none]

// Definition of class Company
ObjFNames[Company, emps + cars + none]

// Definition of class Insurance
ObjAttrib[Insurance, kind , InsuranceKindEnumCD]
ObjFNames[Insurance, kind + none]

// Definition of class Employee
ObjFNames[Employee, insurances + worksIn + none]

// Definition of class Vehicle
ObjAttrib[Vehicle, licensePlate , type_String]
ObjAttrib[Vehicle, registrationDate , type_Date]
ObjFNames[Vehicle, licensePlate + registrationDate +
        drivenBy + none]

// Special properties of singletons, abstract classes and interfaces
no Vehicle

// All classes in this CD
Obj = Driver + Car + License + Truck + Address + Company +
    Insurance + Employee + Vehicle

// Associations
ObjLUAttrib[EmployeeSubsCD, worksIn, AddressSubsCD, 1, 3]
ObjLU[AddressSubsCD, worksIn, EmployeeSubsCD, 1, 1]
BidiAssoc[DriverSubsCD, drives, VehicleSubsCD, drivenBy]
ObjLUAttrib[VehicleSubsCD, drivenBy, DriverSubsCD, 1, 1]
ObjLUAttrib[DriverSubsCD, drives, VehicleSubsCD, 0, 1]
ObjLUAttrib[EmployeeSubsCD, insurances, InsuranceSubsCD, 1]
Listing 3.1: The Alloy module generated from class diagram \( cd_1 \) presented in Sect. 1.2 (Listing 1.3 and Fig. 1.1).

The Alloy code from Listing 3.1 can be executed in the Alloy Analyzer together with the foundational part of the Alloy module from Listing 2.1 and a run command. The run command as used in the CD2Alloy tool with scope 10 is `run cd for 10`. 
Chapter 4

Translation of Alloy Instances Back to ODs

Intermediate results of the CD analysis using CD2Alloy are Alloy instances of the Alloy modules generated from input CDs. These Alloy instances represent object models of the CDs (Alloy instances in scope $k$ exist iff object models with at most $k$ objects exist). The Alloy instances can be presented to the engineer as witnesses of the analysis result by a translation to object diagrams.

4.1 Structure of CD2Alloy Alloy instances

The Alloy instances generated by an execution of the Alloy Analyzer on Alloy modules generated by CD2Alloy represent object models of the input CDs. These instances have similar structure due to the generic foundational part of the CD2Alloy translation. The relevant parts of the generated Alloy instances are:

- **Obj** a set of atoms representing all objects in the OM
- **Val** a set of atoms representing values of primitive or unknown types of attributes in the OM
- **EnumVal** a set of atoms representing enumeration values assigned to attributes in the OM
- **FName** a set of atoms representing attribute and role names
- **Obj<:get** a set of tuples of type $\text{Obj} \times \text{FName} \times (\text{Obj} \cup \text{Val} \cup \text{EnumVal})$ representing links and assignments to attributes in the OM
The set \( \text{univ} \) is the set of all atoms in an Alloy instance. We use the function \( \text{sig} : \text{univ} \rightarrow \text{Name} \) to retrieve the most specific signature name of an atom. We use the function \( \text{name} : \text{univ} \rightarrow \text{Name} \) to retrieve the name of an atom (Alloy names atoms after their most specific signature and appends a running number per signature).

### 4.2 Translation rules

CD2Alloy translates Alloy instances representing object models to UML/P object diagrams. The UML/P ODs used in CD2Alloy consist of objects with attributes and links between these objects (the MontiCore grammar for object diagrams can be found in [Sch12]). Alloy instances are translated to ODs according to the rules in Fig. 4.1.

```plaintext
package temp;

objectdiagram of {
  \forall o \in \text{Obj}:
    o.name : o.sig {
      1 \forall (o', \text{name}, \text{val}) \in \text{Obj:<:get} \text{ s.t. } o' = o \land \text{val} \in \text{Val}:
          \text{val.sig} \text{name}.name = \text{some.val.sig};
      2 \forall (o', \text{name}, eVal) \in \text{Obj:<:get} \text{ s.t. } o' = o \land e\text{Val} \in \text{EnumVal}:
          \text{enumName}(\text{evVal.name}) \text{name}.name = \text{enumValueName}(\text{evVal.name});
    }
  3 \forall (o, \text{rname}, o') \in \text{Obj:<:get} \text{ s.t. } o, o' \in \text{Obj}:
      \text{link} o.name \rightarrow (\text{rname.name}) o'.name;
}
```

Figure 4.1: Translation of CD2Alloy Alloy instances to UML/P ODs consisting of objects with attributes and links between them.

Rule 1 in Fig. 4.1 creates the OD syntax declaration of an object for every atom in \( \text{Obj} \). Inside this declaration rules 2 and 3 add the object’s attributes. Rule 2 adds attributes of primitive or unknown types (atoms from set \( \text{Val} \)). Names of atoms representing enumeration values have the format `enum_enumName_enumValueName`. The helper functions `enumName` and `enumValueName` used in rule 3 compute these enumeration names and specific value names for atoms of the set \( \text{EnumVal} \) by syntactic manipulations.
Rule 4 creates the declaration of a link in the object diagram created from the Alloy instance.

### 4.3 Example of CD2Alloy Alloy instance to OD translation

We present an Alloy instance computed from the CD2Alloy translation of the class diagram from Sect. 1.2. The instance shown in Listing 4.1 was computed by the Alloy Analyzer using scope 10.

```alloy
Obj = { Address0, Address1, Car0, Company0, Driver0, Employee0, Insurance0, Insurance1 }

Driver = { Driver0 }
Car = { Car0 }
License = {}
Truck = {}
Address = { Address0, Address1 }
Company = { Company0 }
Employee = { Employee0 }
Insurance = { Insurance0, Insurance1 }
Vehicle = {}

FName = {
  cars, color, drivenBy, drives, drivingExperience, emps, insurances, kind, license, licensePlate, of, ownedBy, registrationDate, worksIn
}

EnumVal = {
  enum_CarKind_regular,
  enum_ColorKind_black,
  enum_DrivingExperience_beginner,
  enum_InsuranceKind_international,
  enum_InsuranceKind_workAccident
}

Val = {type_Date, type_String}

Obj <: get = {
  Car0 -> color -> enum_ColorKind_black,
  Car0 -> drivenBy -> Driver0,
  Car0 -> kind -> enum_CarKind_regular,
  Car0 -> licensePlate -> type_String,
  ...}
```
The CD2Alloy Alloy instance shown in Listing 4.1 is translated to the object diagram shown in Listing 4.2 using the translation rules given in Sect. 4.2.
Listing 4.2: The UML/P OD generated by the translation of the Alloy instance from Listing 4.1.
Bibliography


### Aachener Informatik-Berichte

This is the list of all technical reports since 1987. To obtain copies of reports please consult http://aib.informatik.rwth-aachen.de/ or send your request to: Informatik-Bibliothek, RWTH Aachen, Ahornstr. 55, 52056 Aachen, Email: biblio@informatik.rwth-aachen.de

<table>
<thead>
<tr>
<th>Year</th>
<th>Title</th>
<th>Authors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1987-01</td>
<td>Fachgruppe Informatik: Jahresbericht 1986</td>
<td></td>
</tr>
<tr>
<td>1987-02</td>
<td>David de Frutos Escrig, Klaus Indermark: Equivalence Relations of Non-Deterministic Ianov-Schemes</td>
<td></td>
</tr>
<tr>
<td>1987-03</td>
<td>Manfred Nagl: A Software Development Environment based on Graph Technology</td>
<td></td>
</tr>
<tr>
<td>1987-04</td>
<td>Claus Lewerentz, Manfred Nagl, Bernhard Westfechtel: On Integration Mechanisms within a Graph-Based Software Development Environment</td>
<td></td>
</tr>
<tr>
<td>1987-05</td>
<td>Reinhard Rinn: Über Eingabeanomalien bei verschiedenen Inferenzmodellen</td>
<td></td>
</tr>
<tr>
<td>1987-06</td>
<td>Werner Damm, Gert Döhmen: Specifying Distributed Computer Architectures in AADL*</td>
<td></td>
</tr>
<tr>
<td>1987-08</td>
<td>Manfred Nagl: Set Theoretic Approaches to Graph Grammars</td>
<td></td>
</tr>
<tr>
<td>1987-09</td>
<td>Claus Lewerentz, Andreas Schürr: Experiences with a Database System for Software Documents</td>
<td></td>
</tr>
<tr>
<td>1987-10</td>
<td>Herbert Klaeren, Klaus Indermark: A New Implementation Technique for Recursive Function Definitions</td>
<td></td>
</tr>
<tr>
<td>1987-11</td>
<td>Rita Loogen: Design of a Parallel Programmable Graph Reduction Machine with Distributed Memory</td>
<td></td>
</tr>
<tr>
<td>1987-12</td>
<td>J. Börstler, U. Möncke, R. Wilhelm: Table compression for tree automata</td>
<td></td>
</tr>
<tr>
<td>1988-01</td>
<td>Gabriele Eser, Johannes Rückert, Frank Wagner Gesellschaftliche Aspekte der Informatik</td>
<td></td>
</tr>
<tr>
<td>1988-02</td>
<td>Peter Martini, Otto Spaniol: Token-Passing in High-Speed Backbone Networks for Campus-Wide Environments</td>
<td></td>
</tr>
<tr>
<td>1988-03</td>
<td>Thomas Welzel: Simulation of a Multiple Token Ring Backbone</td>
<td></td>
</tr>
<tr>
<td>1988-04</td>
<td>Peter Martini: Performance Comparison for HSLAN Media Access Protocols</td>
<td></td>
</tr>
<tr>
<td>1988-05</td>
<td>Peter Martini: Performance Analysis of Multiple Token Rings</td>
<td></td>
</tr>
<tr>
<td>1988-06</td>
<td>Andreas Mann, Johannes Rückert, Otto Spaniol: Datenfunknetze</td>
<td></td>
</tr>
<tr>
<td>1988-07</td>
<td>Andreas Mann, Johannes Rückert: Packet Radio Networks for Data Exchange</td>
<td></td>
</tr>
<tr>
<td>1988-08</td>
<td>Andreas Mann, Johannes Rückert: Concurrent Slot Assignment Protocol for Packet Radio Networks</td>
<td></td>
</tr>
<tr>
<td>1988-09</td>
<td>W. Kremer, F. Reichert, J. Rückert, A. Mann: Entwurf einer Netzwerktopologie für ein Mobilfunknetz zur Unterstützung des öffentlichen Straßenverkehrs</td>
<td></td>
</tr>
<tr>
<td>1988-10</td>
<td>Kai Jakobs: Towards User-Friendly Networking</td>
<td></td>
</tr>
<tr>
<td>1988-12</td>
<td>Kai Jakobs: Directory Services in Distributed Systems - A Survey</td>
<td></td>
</tr>
<tr>
<td>1988-13</td>
<td>Martine Schümmer: RS-511, a Protocol for the Plant Floor</td>
<td></td>
</tr>
<tr>
<td>Year</td>
<td>Title</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-----------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>1988</td>
<td>Fachgruppe Informatik: Jahresbericht 1987</td>
<td></td>
</tr>
<tr>
<td>1988</td>
<td>Wolfgang Thomas: Automata on Infinite Objects</td>
<td></td>
</tr>
<tr>
<td>1988</td>
<td>Heiko Vogler: Functional Distribution of the Contextual Analysis in Block-Structured Programming Languages: A Case Study of Tree Transducers</td>
<td></td>
</tr>
<tr>
<td>1988</td>
<td>Joost Engelfriet, Heiko Vogler: Modular Tree Transducers</td>
<td></td>
</tr>
<tr>
<td>1988</td>
<td>Wolfgang Thomas: Automata and Quantifier Hierarchies</td>
<td></td>
</tr>
<tr>
<td>1988</td>
<td>Uschi Heuter: Generalized Definite Tree Languages</td>
<td></td>
</tr>
<tr>
<td>1988</td>
<td>Fachgruppe Informatik: Jahresbericht 1988</td>
<td></td>
</tr>
<tr>
<td>1989</td>
<td>G. Esser, J. Rückert, F. Wagner (Hrsg.): Gesellschaftliche Aspekte der Informatik</td>
<td></td>
</tr>
<tr>
<td>1989</td>
<td>Heiko Vogler: Bottom-Up Computation of Primitive Recursive Tree Functions</td>
<td></td>
</tr>
<tr>
<td>1989</td>
<td>Andy Schürr: Introduction to PROGRESS, an Attribute Graph Grammar Based Specification Language</td>
<td></td>
</tr>
<tr>
<td>1989</td>
<td>J. Börstler: Reuse and Software Development - Problems, Solutions, and Bibliography (in German)</td>
<td></td>
</tr>
<tr>
<td>1989</td>
<td>Kai Jakobs: OSI - An Appropriate Basis for Group Communication?</td>
<td></td>
</tr>
<tr>
<td>1989</td>
<td>Bernhard Westfechtel: Extension of a Graph Storage for Software Documents with Primitives for Undo/Redo and Revision Control</td>
<td></td>
</tr>
<tr>
<td>1989</td>
<td>Peter Martini: High Speed Local Area Networks - A Tutorial</td>
<td></td>
</tr>
<tr>
<td>1989</td>
<td>P. Davids, Th. Welzel: Performance Analysis of DQDB Based on Simulation</td>
<td></td>
</tr>
<tr>
<td>1989</td>
<td>Manfred Nagl (Ed.): Abstracts of Talks presented at the WG ’89 15th International Workshop on Graphtheoretic Concepts in Computer Science</td>
<td></td>
</tr>
<tr>
<td>1989</td>
<td>Peter Martini: The DQDB Protocol - Is it Playing the Game?</td>
<td></td>
</tr>
<tr>
<td>1989</td>
<td>Martine Schümer: CNC/DNC Communication with MAP</td>
<td></td>
</tr>
<tr>
<td>1989</td>
<td>Martine Schümer: Local Area Networks for Manufacturing Environments with hard Real-Time Requirements</td>
<td></td>
</tr>
<tr>
<td>1989</td>
<td>G. Vossen, K.-U. Witt: SUXESS: Towards a Sound Unification of Extensions of the Relational Data Model</td>
<td></td>
</tr>
</tbody>
</table>
1989-18 A. Maassen: Programming with Higher Order Functions
1990-01 ∗ Fachgruppe Informatik: Jahresbericht 1989
1990-02 ∗ Vera Jansen, Andreas Potthoff, Wolfgang Thomas, Udo Wermuth: A Short Guide to the AMORE System (Computing Automata, MOnoids and Regular Expressions)
1990-03 ∗ Jerzy Skurczynski: On Three Hierarchies of Weak SkS Formulas
1990-04 R. Loogen: Stack-based Implementation of Narrowing
1990-05 H. Kuchen, A. Wagener: Comparison of Dynamic Load Balancing Strategies
1990-06 ∗ Kai Jakobs, Frank Reichert: Directory Services for Mobile Communication
1990-07 ∗ Kai Jakobs: What’s Beyond the Interface - OSI Networks to Support Cooperative Work
1990-08 ∗ Kai Jakobs: Directory Names and Schema - An Evaluation
1990-09 ∗ Ulrich Quernheim, Dieter Kreuer: Das CCITT - Signalisierungssystem Nr. 7 auf Satellitenstrecken; Simulation der Zeichengabestrecke
1990-12 ∗ Kai Jakobs, Josef Kaltwasser, Frank Reichert, Otto Spaniol: Der Computer fährt mit
1990-13 ∗ Rudolf Mathar, Andreas Mann: Analyzing a Distributed Slot Assignment Protocol by Markov Chains
1990-14 A. Maassen: Compilerentwicklung in Miranda - ein Praktikum in funktionaler Programmierung (written in german)
1990-15 ∗ Manfred Nagl, Andreas Schürr: A Specification Environment for Graph Grammars
1990-16 A. Schürr: PROGRESS: A VHL-Language Based on Graph Grammars
1990-17 ∗ Marita Möller: Ein Ebenenmodell wissensbasierter Konsultationen - Unterstützung für Wissensakquisition und Erklärungsfähigkeit
1990-18 ∗ Eric Kowalewski: Entwurf und Interpretation einer Sprache zur Beschreibung von Konsultationsphasen in Expertensystemen
1990-21 ∗ Manfred Nagl: Modelling of Software Architectures: Importance, Notions, Experiences
1991-03 B. Steffen, A. Ingolfsdottir: Characteristic Formulae for Processes with Divergence
1991-04 M. Portz: A new class of cryptosystems based on interconnection networks
1991-05  H. Kuchen, G. Geiler: Distributed Applicative Arrays
1991-06  * Ludwig Staiger: Kolmogorov Complexity and Hausdorff Dimension
1991-07  * Ludwig Staiger: Syntactic Congruences for w-languages
1991-09  * Eila Kuikkka: A Proposal for a Syntax-Directed Text Processing System
1991-11  R. Loogen, St. Winkler: Dynamic Detection of Determinism in Functional Logic Languages
1991-14  * Andreas Fasbender, Wolfgang Kremer: A New Approximation Algorithm for Tandem Networks with Priority Nodes
1991-15  J. Börstler, A. Zündorf: Revisiting extensions to Modula-2 to support reusability
1991-16  J. Börstler, Th. Janning: Bridging the gap between Requirements Analysis and Design
1991-17  A. Zündorf, A. Schürr: Nondeterministic Control Structures for Graph Rewriting Systems
1991-19  M. Jeusfeld, M. Jarke: From Relational to Object-Oriented Integrity Simplification
1991-20  G. Hogen, A. Kindler, R. Loogen: Automatic Parallelization of Lazy Functional Programs
1991-21  * Prof. Dr. rer. nat. Otto Spaniol: ODP (Open Distributed Processing): Yet another Viewpoint
1991-23  S. Graf, B. Steffen: Compositional Minimization of Finite State Systems
1991-27  J. Knoop, B. Steffen: The Interprocedural Coincidence Theorem
1991-28  J. Knoop, B. Steffen: Unifying Strength Reduction and Semantic Code Motion
1991-30  T. Margaria: First-Order theories for the verification of complex FSMs
1991-31  B. Steffen: Generating Data Flow Analysis Algorithms from Modal Specifications
1992-02  * Bernhard Westfechtel: Basismechanismen zur Datenverwaltung in strukturbezogenen Hypertextsystemen
1992-04  S. A. Smolka, B. Steffen: Priority as Extremal Probability
1992-05  * Matthias Jarke, Carlos Maltzahn, Thomas Rose: Sharing Processes: Team Coordination in Design Repositories
<table>
<thead>
<tr>
<th>Year</th>
<th>Authors</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1992-06</td>
<td>O. Burkart, B. Steffen</td>
<td>Model Checking for Context-Free Processes</td>
</tr>
<tr>
<td>1992-07</td>
<td>Matthias Jarke, Klaus Pohl</td>
<td>Information Systems Quality and Quality Information Systems</td>
</tr>
<tr>
<td>1992-08</td>
<td>Rudolf Mathar, Jürgen Mattfeldt</td>
<td>Analyzing Routing Strategy NFP in Multihop Packet Radio Networks on a Line</td>
</tr>
<tr>
<td>1992-09</td>
<td>Alfons Kemper, Guido Moerkotte</td>
<td>Grundlagen objektorientierter Datenbanksysteme</td>
</tr>
<tr>
<td>1992-10</td>
<td>Matthias Jarke, Manfred Jeusfeld, Andreas Miethsam, Michael Gocek</td>
<td>Towards a logic-based reconstruction of software configuration management</td>
</tr>
<tr>
<td>1992-11</td>
<td>Werner Hans</td>
<td>A Complete Indexing Scheme for WAM-based Abstract Machines</td>
</tr>
<tr>
<td>1992-12</td>
<td>W. Hans, R. Loogen, St. Winkler</td>
<td>On the Interaction of Lazy Evaluation and Backtracking</td>
</tr>
<tr>
<td>1992-13</td>
<td>Matthias Jarke, Thomas Rose</td>
<td>Specification Management with CAD</td>
</tr>
<tr>
<td>1992-15</td>
<td>A. Schuerr, B. Westfechtel</td>
<td>Graphgrammatiken und Graphersetzungssysteme (written in german)</td>
</tr>
<tr>
<td>1992-16</td>
<td>Graduiertenkolleg Informatik und Technik (Hrsg.)</td>
<td>Forschungsprojekte des Graduiertenkolleges Informatik und Technik</td>
</tr>
<tr>
<td>1992-18</td>
<td>Clarence A. Ellis, Matthias Jarke (Eds.)</td>
<td>Distributed Cooperation in Integrated Information Systems - Proceedings of the Third International Workshop on Intelligent and Cooperative Information Systems</td>
</tr>
<tr>
<td>1992-19-00</td>
<td>H. Kuchen, R. Loogen (eds.)</td>
<td>Proceedings of the 4th Int. Workshop on the Parallel Implementation of Functional Languages</td>
</tr>
<tr>
<td>1992-19-01</td>
<td>G. Hogen, R. Loogen</td>
<td>PASTEL - A Parallel Stack-Based Implementation of Eager Functional Programs with Lazy Data Structures (Extended Abstract)</td>
</tr>
<tr>
<td>1992-19-03</td>
<td>C. Rathsack, S.B. Scholz</td>
<td>LISA - A Lazy Interpreter for a Full-Fledged Lambda-Calculus</td>
</tr>
<tr>
<td>1992-19-04</td>
<td>T.A. Bratvold</td>
<td>Determining Useful Parallelism in Higher Order Functions</td>
</tr>
<tr>
<td>1992-19-05</td>
<td>S. Kahrs</td>
<td>Polymorphic Type Checking by Interpretation of Code</td>
</tr>
<tr>
<td>1992-19-09</td>
<td>D. Howe, G. Burn</td>
<td>Experiments with strict STG code</td>
</tr>
<tr>
<td>1992-19-12</td>
<td>A. Bennett, P. Kelly</td>
<td>Simulation of Multicache Parallel Reduction</td>
</tr>
</tbody>
</table>
1992-19-19 D. Lester: Distributed Garbage Collection of Cyclic Structures (Draft version)
1992-19-27 H. Doerr: Monitoring with Graph-Grammars as formal operational Models
1992-19-28 J. van Groningen: Some implementation aspects of Concurrent Clean on distributed memory architectures
1992-23 T. Margaria, B. Steffen: Distinguishing Formulas for Free
1992-27 W. Hans, St. Winkler: Aliasing and Groundness Analysis of Logic Programs through Abstract Interpretation and its Safety
1992-29 B. Schinzel: Warum Frauenforschung in Naturwissenschaft und Technik
1992-32 * Bernd Heinrichs, Kai Jakobs: Timer Handling in High-Performance Transport Systems
1992-34 C. Gerlhof, A. Kemper, Ch. Kilger, G. Moerkotte: Partition-Based Clustering in Object Bases: From Theory to Practice
1992-35 J. Börstler: Feature-Oriented Classification and Reuse in IPSEN
1992-38 A. Zuendorf: Implementation of the imperative / rule based language PROGRES
1992-40 * Rudolf Mathar, Jürgen Mattfeldt: Channel Assignment in Cellular Radio Networks
1992-41 * Gerhard Friedrich, Wolfgang Neidl: Constructive Utility in Model-Based Diagnosis Repair Systems
1992-42 * P. S. Chen, R. Hennicker, M. Jarke: On the Retrieval of Reusable Software Components
1992-43 W. Hans, St.Winkler: Abstract Interpretation of Functional Logic Languages
1992-44 N. Kiesel, A. Schuerr, B. Westfechtel: Design and Evaluation of GRAS, a Graph-Oriented Database System for Engineering Applications
1993-01 * Fachgruppe Informatik: Jahresbericht 1992
1993-02 * Patrick Shi Cheng Chen: On Inference Rules of Logic-Based Information Retrieval Systems
1993-03 G. Hogen, R. Loogen: A New Stack Technique for the Management of Runtime Structures in Distributed Environments
1993-05 A. Zündorf: A Heuristic for the Subgraph Isomorphism Problem in Executing PROGRES
1993-06 A. Kemper, D. Kossmann: Adaptable Pointer Swizzling Strategies in Object Bases: Design, Realization, and Quantitative Analysis
1993-07 * Graduiertenkolleg Informatik und Technik (Hrsg.): Graduiertenkolleg Informatik und Technik
1993-08 * Matthias Berger: k-Coloring Vertices using a Neural Network with Convergence to Valid Solutions
1993-09 M. Buchheit, M. Jeusfeld, W. Nutt, M. Staudt: Subsumption between Queries to Object-Oriented Databases
1993-10 O. Burkart, B. Steffen: Pushdown Processes: Parallel Composition and Model Checking
1993-12 * Rudolf Mathar, Jürgen Mattfeldt: On the Distribution of Cumulated Interference Power in Rayleigh Fading Channels
1993-13 O. Maler, L. Staiger: On Syntactic Congruences for \omega\textquotesingle{}languages
1993-14 M. Jarke, St. Eherer, R. Gallersdoerfer, M. Jeusfeld, M. Staudt: ConceptBase - A Deductive Object Base Manager
1993-17 * M. Jarke, K. Pohl: Establishing Vision in Context: Towards a Model of Requirements Processes
1993-18 W. Hans, H. Kuchen, St. Winkler: Full Indexing for Lazy Narrowing
1993-21 M. Jarke, M.A. Jeusfeld, P. Scezurko: Three Aspects of Intelligent Co-operation in the Quality Cycle
1994-01 Margit Generet, Sven Martin (eds.), Fachgruppe Informatik: Jahresbericht 1993
1994-04 * Rolf Hager, Rudolf Mathar, Jürgen Mattfeldt: Intelligent Cruise Control and Reliable Communication of Mobile Stations
1994-05 * Rolf Hager, Peter Hermesmann, Michael Portz: Feasibility of Authentication Procedures within Advanced Transport Telematics
1994-07 P. Peters, P. Szczurko: Integrating Models of Quality Management Methods by an Object-Oriented Repository
1994-08 * Manfred Nagl, Bernhard Westfechtel: A Universal Component for the Administration in Distributed and Integrated Development Environments
1994-09 * Patrick Horster, Holger Petersen: Signatur- und Authentifikationsverfahren auf der Basis des diskreten Logarithmusproblems
1994-11 A. Schürr: PROGRES, A Visual Language and Environment for Programming with Graph REwrite Systems
1994-12 A. Schürr: Specification of Graph Translators with Triple Graph Grammars
1994-13 A. Schürr: Logic Based Programmed Structure Rewriting Systems
1994-14 L. Staiger: Codes, Simplifying Words, and Open Set Condition
1994-15 * Bernhard Westfechtel: A Graph-Based System for Managing Configurations of Engineering Design Documents
1994-16 P. Klein: Designing Software with Modula-3
1994-17 I. Litovsky, L. Staiger: Finite acceptance of infinite words
1994-18 G. Hogen, R. Loogen: Parallel Functional Implementations: Graphbased vs. Stackbased Reduction
1994-20 * R. Gallersdörfer, M. Jarke, K. Klabunde: Intelligent Networks as a Data Intensive Application (INDIA)
1994-21 M. Mohnen: Proving the Correctness of the Static Link Technique Using Evolving Algebras
1994-22 H. Fernau, L. Staiger: Valuations and Unambiguity of Languages, with Applications to Fractal Geometry
1994-26 * St. Jacobs, St. Kethers: Improving Communication and Decision Making within Quality Function Deployment
1995-01 * Fachgruppe Informatik: Jahresbericht 1994
1995-02 Andy Schürr, Andreas J. Winter, Albert Zündorf: Graph Grammar Engineering with PROGRES
1995-03 Ludwig Staiger: A Tight Upper Bound on Kolmogorov Complexity by Hausdorff Dimension and Uniformly Optimal Prediction
1995-04 Birgitta König-Ries, Sven Helmer, Guido Moerkotte: An experimental study on the complexity of left-deep join ordering problems for cyclic queries
1995-05 Sophie Cluet, Guido Moerkotte: Efficient Evaluation of Aggregates on Bulk Types
1995-06 Sophie Cluet, Guido Moerkotte: Nested Queries in Object Bases
1995-07 Sophie Cluet, Guido Moerkotte: Query Optimization Techniques Exploiting Class Hierarchies
1995-08 Markus Mohnen: Efficient Compile-Time Garbage Collection for Arbitrary Data Structures
1995-09 Markus Mohnen: Functional Specification of Imperative Programs: An Alternative Point of View of Functional Languages
1995-10 Rainer Gallersdörfer, Matthias Nicola: Improving Performance in Replicated Databases through Relaxed Coherency
1995-11 * M.Staudt, K.von Thadden: Subsumption Checking in Knowledge Bases
1995-12 * G.V.Zemanek, H.W.Nissen, H.Hubert, M.Jarke: Requirements Analysis from Multiple Perspectives: Experiences with Conceptual Modeling Technology
1995-14 * P.Peters, P.Szczurko, M.Jeusfeld: Oriented Information Management: Conceptual Models at Work
1995-15 * Matthias Jarke, Sudha Ram (Hrsg.): WITS 95 Proceedings of the 5th Annual Workshop on Information Technologies and Systems

1995-16 * W.Hans, St.Winkler, F Saenz: Distributed Execution in Functional Logic Programming

1996-01 * Jahresbericht 1995


1996-03 * W.Scheufele, G.Moerkotte: Optimal Ordering of Selections and Joins in Acyclic Queries with Expensive Predicates

1996-04 Klaus Pohl: PRO-ART: Enabling Requirements Pre-Traceability

1996-05 Klaus Pohl: Requirements Engineering: An Overview


1996-07 Olaf Chitil: The Sigma-Semantics: A Comprehensive Semantics for Functional Programs

1996-08 * S.Sripada: On Entropy and the Limitations of the Second Law of Thermodynamics

1996-09 Michael Hanus (Ed.): Proceedings of the Poster Session of ALP96 - Fifth International Conference on Algebraic and Logic Programming

1996-09-0 Michael Hanus (Ed.): Proceedings of the Poster Session of ALP 96 - Fifth International Conference on Algebraic and Logic Programming: Introduction and table of contents

1996-09-1 Ilies Alouini: An Implementation of Conditional Concurrent Rewriting on Distributed Memory Machines

1996-09-2 Olivier Danvy, Karoline Malmkjær: On the Idempotence of the CPS Transformation

1996-09-3 Víctor M. Gulías, José L. Freire: Concurrent Programming in Haskell

1996-09-4 Sébastien Limet, Pierre Réty: On Decidability of Unifiability Modulo Rewrite Systems

1996-09-5 Alexandre Tessier: Declarative Debugging in Constraint Logic Programming


1996-11 * C.Weise, D.Lenzkes: A Fast Decision Algorithm for Timed Refinement


1996-16 * M.Jarke, M.Gebhardt, S.Jacobs, H.Nissen: Conflict Analysis Across Heterogeneous Viewpoints: Formalization and Visualization

1996-17 Manfred A. Jeusfeld, Tung X. Bui: Decision Support Components on the Internet
1996-19 ∗ P. Peters, M. Jarke: Simulating the impact of information flows in networked organizations
1996-20 Matthias Jarke, Peter Peters, Manfred A. Jeusfeld: Model-driven planning and design of cooperative information systems
1996-23 ∗ M. Gebhardt, S. Jacobs: Conflict Management in Design
1997-01 Michael Hanus, Frank Zartmann (eds.): Jahresbericht 1996
1997-02 Johannes Faassen: Using full parallel Boltzmann Machines for Optimization
1997-03 Andreas Winter, Andy Schürr: Modules and Updatable Graph Views for PROgrammed Graph REwriting Systems
1997-04 Markus Mohnen, Stefan Tobies: Implementing Context Patterns in the Glasgow Haskell Compiler
1997-05 ∗ S. Gruner: Schemakorrespondenzaxiome unterstützen die paargrammatische Spezifikation inkrementeller Integrationswerkzeuge
1997-06 Matthias Nicola, Matthias Jarke: Design and Evaluation of Wireless Health Care Information Systems in Developing Countries
1997-07 Petra Hofstedt: Taskparallele Skelette für irregulär strukturierte Probleme in deklarativen Sprachen
1997-08 Dorothea Blostein, Andy Schürr: Computing with Graphs and Graph Rewriting
1997-09 Carl-Arndt Krapp, Bernhard Westfechtel: Feedback Handling in Dynamic Task Nets
1997-10 Matthias Nicola, Matthias Jarke: Integrating Replication and Communication in Performance Models of Distributed Databases
1997-13 Markus Mohnen: Optimising the Memory Management of Higher-Order Functional Programs
1997-14 Roland Baumann: Client/Server Distribution in a Structure-Oriented Database Management System
1998-01 ∗ Fachgruppe Informatik: Jahresbericht 1997
1998-03 Stefan Gruner: Einige Anmerkungen zur graphgrammatischen Spezifikation von Integrationswerkzeugen nach Westfechtel, Janning, Lefering und Schürr
1998-04 ∗ O. Kubitz: Mobile Robots in Dynamic Environments
<table>
<thead>
<tr>
<th>Year</th>
<th>Authors</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1998-06</td>
<td>Matthias Oliver Berger</td>
<td>DECT in the Factory of the Future</td>
</tr>
<tr>
<td>1998-09</td>
<td>Th. Lehmann</td>
<td>Geometrische Ausrichtung medizinischer Bilder am Beispiel intraoraler Radiographien</td>
</tr>
<tr>
<td>1998-10</td>
<td>M. Nicola, M. Jarke</td>
<td>Performance Modeling of Distributed and Replicated Databases</td>
</tr>
<tr>
<td>1998-11</td>
<td>Ansgar Schleicher, Bernhard Westfechtel, Dirk Jäger</td>
<td>Modeling Dynamic Software Processes in UML</td>
</tr>
<tr>
<td>1998-13</td>
<td>Klaus Indermark</td>
<td>Semantik rekursiver Funktionsdefinitionen mit Striktheitsinformation</td>
</tr>
<tr>
<td>1999-01</td>
<td></td>
<td>Jahresbericht 1998</td>
</tr>
<tr>
<td>1999-02</td>
<td>F. Huch</td>
<td>Verification of Erlang Programs using Abstract Interpretation and Model Checking — Extended Version</td>
</tr>
<tr>
<td>1999-03</td>
<td>R. Gallersdörfer, M. Jarke, M. Nicola</td>
<td>The ADR Replication Manager</td>
</tr>
<tr>
<td>1999-04</td>
<td>María Alpuente, Michael Hanus, Salvador Lucas, Germán Vidal</td>
<td>Specialization of Functional Logic Programs Based on Needed Narrowing</td>
</tr>
<tr>
<td>1999-05</td>
<td>W. Thomas (Ed.)</td>
<td>DLT 99 - Developments in Language Theory Fourth International Conference</td>
</tr>
<tr>
<td>1999-06</td>
<td>Kai Jakobs, Klaus-Dieter Kleefeld</td>
<td>Informationssysteme für die angewandte historische Geographie</td>
</tr>
<tr>
<td>1999-07</td>
<td>Thomas Wilke</td>
<td>CTL+ is exponentially more succinct than CTL</td>
</tr>
<tr>
<td>1999-08</td>
<td>Oliver Matz</td>
<td>Dot-Depth and Monadic Quantifier Alternation over Pictures</td>
</tr>
<tr>
<td>2000-01</td>
<td></td>
<td>Jahresbericht 1999</td>
</tr>
<tr>
<td>2000-02</td>
<td>Jens Vögé, Marcin Jurdzinski</td>
<td>A Discrete Strategy Improvement Algorithm for Solving Parity Games</td>
</tr>
<tr>
<td>2000-03</td>
<td>D. Jäger, A. Schleicher, B. Westfechtel</td>
<td>UPGRADE: A Framework for Building Graph-Based Software Engineering Tools</td>
</tr>
<tr>
<td>2000-05</td>
<td>Mareike Schoop</td>
<td>Cooperative Document Management</td>
</tr>
<tr>
<td>2000-06</td>
<td>Mareike Schoop, Christoph Quix (eds.)</td>
<td>Proceedings of the Fifth International Workshop on the Language-Action Perspective on Communication Modelling</td>
</tr>
<tr>
<td>2000-07</td>
<td>Markus Mohnen, Pieter Koopman (Eds.)</td>
<td>Proceedings of the 12th International Workshop of Functional Languages</td>
</tr>
<tr>
<td>2000-08</td>
<td>Thomas Arts, Thomas Noll</td>
<td>Verifying Generic Erlang Client-Server Implementations</td>
</tr>
<tr>
<td>2001-01</td>
<td></td>
<td>Jahresbericht 2000</td>
</tr>
<tr>
<td>2001-02</td>
<td>Benedikt Bollig, Martin Leucker</td>
<td>Deciding LTL over Mazurkiewicz Traces</td>
</tr>
<tr>
<td>2001-03</td>
<td>Thierry Cachat</td>
<td>The power of one-letter rational languages</td>
</tr>
</tbody>
</table>
2001-04 Benedikt Bollig, Martin Leucker, Michael Weber: Local Parallel Model Checking for the Alternation Free μ-Calculus
2001-05 Benedikt Bollig, Martin Leucker, Thomas Noll: Regular MSC Languages
2001-06 Achim Blumensath: Prefix-Recognisable Graphs and Monadic Second-Order Logic
2001-07 Martin Grohe, Stefan Wöhre: An Existential Locality Theorem
2001-08 Mareike Schoop, James Taylor (eds.): Proceedings of the Sixth International Workshop on the Language-Action Perspective on Communication Modelling
2001-09 Thomas Arts, Jürgen Giesl: A collection of examples for termination of term rewriting using dependency pairs
2001-10 Achim Blumensath: Axiomatising Tree-interpretable Structures
2001-11 Klaus Indermark, Thomas Noll (eds.): Kolloquium Programmiersprachen und Grundlagen der Programmierung
2002-01 * Jahresbericht 2001
2002-02 Jürgen Giesl, Aart Middeldorp: Transformation Techniques for Context-Sensitive Rewrite Systems
2002-03 Benedikt Bollig, Martin Leucker, Thomas Noll: Generalised Regular MSC Languages
2002-04 Jürgen Giesl, Aart Middeldorp: Innermost Termination of Context-Sensitive Rewriting
2002-05 Horst Lichter, Thomas von der Maßen, Thomas Weiler: Modelling Requirements and Architectures for Software Product Lines
2002-06 Henry N. Adorna: 3-Party Message Complexity is Better than 2-Party Ones for Proving Lower Bounds on the Size of Minimal Nondeterministic Finite Automata
2002-07 Jörg Dahmen: Invariant Image Object Recognition using Gaussian Mixture Densities
2002-08 Markus Mohnen: An Open Framework for Data-Flow Analysis in Java
2002-09 Markus Mohnen: Interfaces with Default Implementations in Java
2002-10 Martin Leucker: Logics for Mazurkiewicz traces
2002-11 Jürgen Giesl, Hans Zantema: Liveness in Rewriting
2003-01 * Jahresbericht 2002
2003-02 Jürgen Giesl, René Thiemann: Size-Change Termination for Term Rewriting
2003-03 Jürgen Giesl, Deepak Kapur: Deciding Inductive Validity of Equations
2003-04 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, Stephan Falke: Improving Dependency Pairs
2003-05 Christof Löding, Philipp Rohde: Solving the Sabotage Game is PSPACE-hard
2003-07 Horst Lichter, Thomas von der Maßen, Alexander Nyßen, Thomas Weiler: Vergleich von Ansätzen zur Feature Modellierung bei der Softwareproduktlinienentwicklung
2003-08 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, Stephan Falke: Mechanizing Dependency Pairs
2004-01 * Fachgruppe Informatik: Jahresbericht 2003
2004-02 Benedikt Bollig, Martin Leucker: Message-Passing Automata are expressively equivalent to EMSO logic
2004-04 Slim Abdennader, Christophe Ringeissen (eds.): RULE 04 – Fifth International Workshop on Rule-Based Programming
2004-05 Herbert Kuchen (ed.): WFLP 04 – 13th International Workshop on Functional and (Constraint) Logic Programming
2004-06 Sergio Antoy, Yoshihito Toyama (eds.): WRS 04 – 4th International Workshop on Reduction Strategies in Rewriting and Programming
2004-07 Michael Codish, Aart Middeldorp (eds.): WST 04 – 7th International Workshop on Termination
2004-08 Klaus Indermark, Thomas Noll: Algebraic Correctness Proofs for Compiling Recursive Function Definitions with Strictness Information
2004-09 Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith: Parameterized Power Domination Complexity
2005-01 * Fachgruppe Informatik: Jahresbericht 2004
2005-03 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp: Proving and Disproving Termination of Higher-Order Functions
2005-04 Daniel Mölle, Stefan Richter, Peter Rossmanith: A Faster Algorithm for the Steiner Tree Problem
2005-05 Fabien Pouget, Thorsten Holz: A Pointillist Approach for Comparing Honeypots
2005-06 Simon Fischer, Berthold Vöcking: Adaptive Routing with Stale Information
2005-07 Felix C. Freiling, Thorsten Holz, Georg Wicherski: Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks
2005-08 Joachim Kneis, Peter Rossmanith: A New Satisfiability Algorithm With Applications To Max-Cut
2005-09 Klaus Kursawe, Felix C. Freiling: Byzantine Fault Tolerance on General Hybrid Adversary Structures
2005-10 Benedikt Bollig: Automata and Logics for Message Sequence Charts
2005-14 Felix C. Freiling, Sukumar Ghosh: Code Stabilization
2005-15 Uwe Naumann: The Complexity of Derivative Computation
2005-16 Uwe Naumann: Syntax-Directed Derivative Code (Part I: Tangent-Linear Code)

2005-17 Uwe Naumann: Syntax-directed Derivative Code (Part II: Intraprocedural Adjoint Code)


2005-19 Uwe Naumann, Andre Vehreschild: Tangent-Linear Code by Augmented LL-Parsers


2005-23 Heiner Ackermann, Alantha Newman, Heiko Röglin, Berthold Vöcking: Decision Making Based on Approximate and Smoothed Pareto Curves


2006-01 * Fachgruppe Informatik: Jahresbericht 2005

2006-02 Michael Weber: Parallel Algorithms for Verification of Large Systems

2006-03 Michael Maier, Uwe Naumann: Intraprocedural Adjoint Code Generated by the Differentiation-Enabled NAGWare Fortran Compiler

2006-04 Ebadollah Varnik, Uwe Naumann, Andrew Lyons: Toward Low Static Memory Jacobian Accumulation

2006-05 Uwe Naumann, Jean Utke, Patrick Heimbach, Chris Hill, Derya Ozyurt, Carl Wunsch, Mike Fagan, Nathan Tallent, Michelle Strout: Adjoint Code by Source Transformation with OpenAD/F

2006-06 Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith: Divide-and-Color

2006-07 Thomas Colcombet, Christof Löding: Transforming structures by set interpretations

2006-08 Uwe Naumann, Yuxiao Hu: Optimal Vertex Elimination in Single-Expression-Use Graphs

2006-09 Tingting Han, Joost-Pieter Katoen: Counterexamples in Probabilistic Model Checking

2006-10 Mesut Günes, Alexander Zimmermann, Martin Wenig, Jan Ritzerfeld, Ulrich Meis: From Simulations to Testbeds - Architecture of the Hybrid MCG-Mesh Testbed

2006-11 Bastian Schlich, Michael Rohrbach, Michael Weber, Stefan Kowalewski: Model Checking Software for Microcontrollers

2006-13  Wong Karianto, Christof Löding: Unranked Tree Automata with Sibling Equalities and Disequalities
2006-15  Sebastian Ullrich, Jakob T. Valvoda, Torsten Kuhlen: Utilizing optical sensors from mice for new input devices
2006-16  Rafael Ballagas, Jan Borchers: Selexels: a Conceptual Framework for Pointing Devices with Low Expressiveness
2006-17  Eric Lee, Henning Kiel, Jan Borchers: Scrolling Through Time: Improving Interfaces for Searching and Navigating Continuous Audio Timelines
2007-01  Fachgruppe Informatik: Jahresbericht 2006
2007-02  Carsten Fuhs, Jürgen Giesl, Aart Middeldorp, Peter Schneider-Kamp, René Thiemann, and Harald Zankl: SAT Solving for Termination Analysis with Polynomial Interpretations
2007-03  Jürgen Giesl, René Thiemann, Stephan Swiderski, and Peter Schneider-Kamp: Proving Termination by Bounded Increase
2007-04  Jan Buchholz, Eric Lee, Jonathan Klein, and Jan Borchers: coJIVE: A System to Support Collaborative Jazz Improvisation
2007-05  Uwe Naumann: On Optimal DAG Reversal
2007-06  Joost-Pieter Katoen, Thomas Noll, and Stefan Rieger: Verifying Concurrent List-Manipulating Programs by LTL Model Checking
2007-07  Alexander Nyßen, Horst Lichter: MeDUSA - MethoD for UML2-based Design of Embedded Software Applications
2007-08  Falk Salewski and Stefan Kowalewski: Achieving Highly Reliable Embedded Software: An empirical evaluation of different approaches
2007-09  Tina Kraußer, Heiko Mantel, and Henning Sudbrock: A Probabilistic Justification of the Combining Calculus under the Uniform Scheduler Assumption
2007-10  Martin Neuhäuser, Joost-Pieter Katoen: Bisimulation and Logical Preservation for Continuous-Time Markov Decision Processes
2007-12  Uwe Naumann: An L-Attributed Grammar for Adjoint Code
2007-13  Uwe Naumann, Michael Maier, Jan Riehme, and Bruce Christianson: Second-Order Adjoints by Source Code Manipulation of Numerical Programs
2007-14  Jean Utke, Uwe Naumann, Mike Fagan, Nathan Tallent, Michelle Strout, Patrick Heimbach, Chris Hill, and Carl Wunsch: OpenAD/F: A Modular, Open-Source Tool for Automatic Differentiation of Fortran Codes
2007-15  Volker Stolz: Temporal assertions for sequential and concurrent programs
2007-17  René Thiemann: The DP Framework for Proving Termination of Term Rewriting
2007-18  Uwe Naumann: Call Tree Reversal is NP-Complete
2007-19 Jan Riehme, Andrea Walther, Jörg Stiller, Uwe Naumann: Adjoints for Time-Dependent Optimal Control
2007-20 Joost-Pieter Katoen, Daniel Klink, Martin Leucker, and Verena Wolf: Three-Valued Abstraction for Probabilistic Systems
2007-21 Tingting Han, Joost-Pieter Katoen, and Alexandru Mereacre: Compositional Modeling and Minimization of Time-Inhomogeneous Markov Chains
2007-22 Heiner Ackermann, Paul W. Goldberg, Vahab S. Mirrokni, Heiko Röglin, and Berthold Vöcking: Uncoordinated Two-Sided Markets
2008-01 Fachgruppe Informatik: Jahresbericht 2007
2008-02 Henrik Bohnenkamp, Marielle Stoelinga: Quantitative Testing
2008-03 Carsten Fuhs, Jürgen Giesl, Aart Middeldorp, Peter Schneider-Kamp, René Thiemann, Harald Zankl: Maximal Termination
2008-04 Uwe Naumann, Jan Riehme: Sensitivity Analysis in Sisyphe with the AD-Enabled NAGWare Fortran Compiler
2008-05 Frank G. Radmacher: An Automata Theoretic Approach to the Theory of Rational Tree Relations
2008-06 Uwe Naumann, Laurent Hascoet, Chris Hill, Paul Hovland, Jan Riehme, Jean Utke: A Framework for Proving Correctness of Adjoint Message Passing Programs
2008-08 George B. Mertzios, Stavros D. Nikolopoulos: The λ-cluster Problem on Parameterized Interval Graphs
2008-09 George B. Mertzios, Walter Unger: An optimal algorithm for the k-fixed-endpoint path cover on proper interval graphs
2008-10 George B. Mertzios, Walter Unger: Preemptive Scheduling of Equal-Length Jobs in Polynomial Time
2008-12 Joost-Pieter Katoen, Daniel Klink, Martin Leucker, Verena Wolf: Abstraction for stochastic systems by Erlang’s method of stages
2008-14 Bastian Schlich: Model Checking of Software for Microcontrollers
2008-16 Hendrik vom Lehn, Elias Weingärtner and Klaus Wehrle: Comparing recent network simulators: A performance evaluation study
2008-17 Peter Schneider-Kamp: Static Termination Analysis for Prolog using Term Rewriting and SAT Solving
2009-02 Taolue Chen, Tingting Han, Joost-Pieter Katoen, Alexandru Mereacre: Quantitative Model Checking of Continuous-Time Markov Chains Against Timed Automata Specifications
2009-03 Alexander Nyßen: Model-Based Construction of Embedded Real-Time Software - A Methodology for Small Devices
2009-04 Daniel Klünder: Entwurf eingebetteter Software mit abstrakten Zustandsmaschinen und Business Object Notation
2009-05 George B. Mertzios, Ignasi Sau, Shmuel Zaks: A New Intersection Model and Improved Algorithms for Tolerance Graphs
2009-06 George B. Mertzios, Ignasi Sau, Shmuel Zaks: The Recognition of Tolerance and Bounded Tolerance Graphs is NP-complete
2009-07 Joachim Kneis, Alexander Langer, Peter Rossmanith: Derandomizing Non-uniform Color-Coding I
2009-08 Joachim Kneis, Alexander Langer: Satellites and Mirrors for Solving Independent Set on Sparse Graphs
2009-09 Michael Nett: Implementation of an Automated Proof for an Algorithm Solving the Maximum Independent Set Problem
2009-10 Felix Reidl, Fernando Sánchez Villaamil: Automatic Verification of the Correctness of the Upper Bound of a Maximum Independent Set Algorithm
2009-11 Kyriaki Ioannidou, George B. Mertzios, Stavros D. Nikolopoulos: The Longest Path Problem is Polynomial on Interval Graphs
2009-12 Martin Neuhäuser, Lijun Zhang: Time-Bounded Reachability in Continuous-Time Markov Decision Processes
2009-13 Martin Zimmermann: Time-optimal Winning Strategies for Poset Games
2009-14 Ralf Huuck, Gerwin Klein, Bastian Schlich (eds.): Doctoral Symposium on Systems Software Verification (DS SSV’09)
2009-15 Joost-Pieter Katoen, Daniel Klink, Martin Neuhäuser: Compositional Abstraction for Stochastic Systems
2009-16 George B. Mertzios, Derek G. Corneil: Vertex Splitting and the Recognition of Trapezoid Graphs
2009-17 Carsten Kern: Learning Communicating and Nondeterministic Automata
2009-18 Paul Hänsch, Michaela Slaats, Wolfgang Thomas: Parametrized Regular Infinite Games and Higher-Order Pushdown Strategies
2010-02 Daniel Neider, Christof Löding: Learning Visibly One-Counter Automata in Polynomial Time
2010-03 Holger Krahn: MontiCore: Agile Entwicklung von domänenspezifischen Sprachen im Software-Engineering
2010-04 René Wörzberger: Management dynamischer Geschäftsprozesse auf Basis statischer Prozessmanagementsysteme
2010-05 Daniel Retkowitz: Softwareunterstützung für adaptive eHome-Systeme
2010-06 Taolue Chen, Tingting Han, Joost-Pieter Katoen, Alexandru Mereacre: Computing maximum reachability probabilities in Markovian timed automata
2010-07 George B. Mertzios: A New Intersection Model for Multitolerance Graphs, Hierarchy, and Efficient Algorithms
2010-08 Carsten Otto, Marc Brockschmidt, Christian von Essen, Jürgen Giesl: Automated Termination Analysis of Java Bytecode by Term Rewriting
2010-09 George B. Mertzios, Shmuel Zaks: The Structure of the Intersection of Tolerance and Cocomparability Graphs
2010-10 Peter Schneider-Kamp, Jürgen Giesl, Thomas Ströder, Alexander Serebrenik, René Thiemann: Automated Termination Analysis for Logic Programs with Cut
2010-11 Martin Zimmermann: Parametric LTL Games
2010-12 Thomas Ströder, Peter Schneider-Kamp, Jürgen Giesl: Dependency Triples for Improving Termination Analysis of Logic Programs with Cut
2010-13 Ashraf Armouch: Design Patterns for Safety-Critical Embedded Systems
2010-14 Michael Codish, Carsten Fuhs, Jürgen Giesl, Peter Schneider-Kamp: Lazy Abstraction for Size-Change Termination
2010-15 Marc Brockschmidt, Carsten Otto, Christian von Essen, Jürgen Giesl: Termination Graphs for Java Bytecode
2010-17 Hans Grönniger: Systemmodell-basierte Definition objektbasierter Modellierungssprachen mit semantischen Variationspunkten
2010-18 Ibrahim Armaç: Personalisierte eHomes: Mobilität, Privatsphäre und Sicherheit
2010-19 Felix Reidl: Experimental Evaluation of an Independent Set Algorithm
2010-20 Wladimir Fridman, Christof Löding, Martin Zimmermann: Degrees of Lookahead in Context-free Infinite Games
2011-02 Marc Brockschmidt, Carsten Otto, Jürgen Giesl: Modular Termination Proofs of Recursive Java Bytecode Programs by Term Rewriting
2011-03 Lars Noschinski, Fabian Emmes, Jürgen Giesl: A Dependency Pair Framework for Innermost Complexity Analysis of Term Rewrite Systems
2011-04 Christina Jansen, Jonathan Heinen, Joost-Pieter Katoen, Thomas Noll: A Local Greibach Normal Form for Hyperedge Replacement Grammars

* These reports are only available as a printed version.
Please contact biblio@informatik.rwth-aachen.de to obtain copies.