Combinatorial Interaction Testing for Test Selection in Grammar-Based Testing

Elke Salecker, Sabine Glesner

Technical University of Berlin, Germany

Workshop on Combinatorial Testing (CT) 2012
Montreal, Canada

Motivation

Grammar-based Testing

- black-box testing approach
- test data derived from context-free grammar

➠ **automatic generation of test data**

Motivation

Grammar-based Testing

- black-box testing approach
- test data derived from context-free grammar

➠ **automatic generation of test data**

Problem

- exhaustive testing of test sets infeasible
- controlled/random-based enumaration process
- rule combination coverage not in focus

Motivation

Grammar-based Testing

- black-box testing approach
- test data derived from context-free grammar

➠ **automatic generation of test data**

Problem

- exhaustive testing of test sets infeasible
- controlled/random-based enumaration process
- rule combination coverage not in focus

New Approach

➠ test selection based on combinatorial interaction testing

➠ automatic generation of CIT test specification

Outline

Combinatorial Interaction Testing

CIT specification: parameters, values

P1	P2	P3	P4
a_1	b_1	c_1	d_1
a_2	b_2	c_2	d_2
a_3			

Combinatorial Interaction Testing

CIT specification: parameters, values

P1	P2	P3	P4
a_1	b_1	c_1	d_1
a_2	b_2	c_2	d_2
a_3			

test case (a_1, b_1, c_1, d_2)

Combinatorial Interaction Testing

CIT specification: parameters, values

P1	P2	P3	P4
a_1	b_1	c_1	d_1
a_2	b_2	c_2	d_2
a_3			

test case (a_1, b_1, c_1, d_2)

test set $P1 \times P2 \times P3 \times P4$, 24 test cases

Combinatorial Interaction Testing

CIT specification: parameters, values

P1	P2	P3	P4
a_1	b_1	c_1	d_1
a_2	b_2	c_2	d_2
a_3			

test case: (a_1, b_1, c_1, d_2)

test set: $P1 \times P2 \times P3 \times P4$, 24 test cases

coverage criterion: (t=2)

1	a_3	b_1	c_1	d_2
2	a_2	b_2	c_2	d_2
3	a_1	b_2	c_1	d_2
4	a_1	b_1	c_2	d_1
5	a_2	b_1	c_1	d_1
6	a_3	b_2	c_2	d_1

Combinatorial Interaction Testing

CIT specification: parameters, values

P1	P2	P3	P4
a_1	b_1	c_1	d_1
a_2	b_2	c_2	d_2
a_3			

test case (a_1, b_1, c_1, d_2)

test set $P1 \times P2 \times P3 \times P4$, 24 test cases

coverage criterion (t=2)

1	a_3	b_1	c_1	d_2
2	a_2	b_2	c_2	d_2
3	a_1	b_2	c_1	d_2
4	a_1	b_1	c_2	d_1
5	a_2	b_1	c_1	d_1
6	a_3	b_2	c_2	d_1

constraints $\neg(b_1 \wedge c_2)$

Grammars

nonterminals $N = \{\epsilon, \mathsf{r}, \mathsf{imm}\}$

terminals $\Sigma = \{\texttt{Const}, \texttt{ObjAddr}, \texttt{Content}, \texttt{Plus}, \texttt{Assign}\}$

Grammars

nonterminals $N = \{\epsilon, \text{r}, \text{imm}\}$

terminals $\Sigma = \{\texttt{Const}, \texttt{ObjAddr}, \texttt{Content}, \texttt{Plus}, \texttt{Assign}\}$

rules (lhs $\longrightarrow$ rhs)

def:	$\epsilon \longrightarrow \texttt{Assign((ObjAddr a), r)}$
add:	$\texttt{r} \longrightarrow \texttt{Plus(r, r)}$
addimm:	$\texttt{r} \longrightarrow \texttt{Plus(r, imm)}$
use:	$\texttt{r} \longrightarrow \texttt{Content(ObjAddr a)}$
r2c:	$\texttt{r} \longrightarrow \texttt{Const c}$
r2i:	$\texttt{r} \longrightarrow \texttt{imm}$
i2c:	$\texttt{imm} \longrightarrow \texttt{Const c}$

start symbol ϵ

Grammars

nonterminals $N = \{\epsilon, \text{r}, \text{imm}\}$

terminals $\Sigma = \{\texttt{Const}, \texttt{ObjAddr}, \texttt{Content}, \texttt{Plus}, \texttt{Assign}\}$

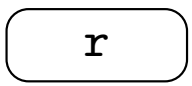
rules (lhs $\longrightarrow$ rhs)

def:	$\epsilon \longrightarrow$ `Assign((ObjAddr a), r)`
add:	`r` $\longrightarrow$ `Plus(r, r)`
addimm:	`r` $\longrightarrow$ `Plus(r, imm)`
use:	`r` $\longrightarrow$ `Content(ObjAddr a)`
r2c:	`r` $\longrightarrow$ `Const c`
r2i:	`r` $\longrightarrow$ `imm`
i2c:	`imm` $\longrightarrow$ `Const c`

start symbol ϵ

derivation sequence of rule applications

$<$ add, addimm, r2c $>$



Grammars

nonterminals $N = \{\epsilon, \text{r}, \text{imm}\}$

terminals $\Sigma = \{\texttt{Const}, \texttt{ObjAddr}, \texttt{Content}, \texttt{Plus}, \texttt{Assign}\}$

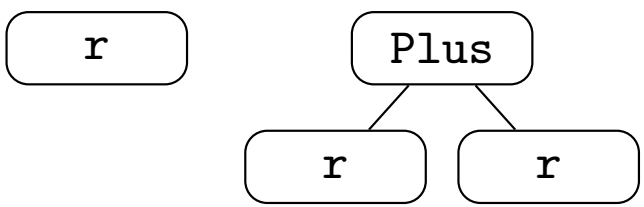
rules (lhs $\rightarrow$ rhs)

def:	$\epsilon \rightarrow \texttt{Assign((ObjAddr a), r)}$
add:	$\texttt{r} \rightarrow \texttt{Plus(r, r)}$
addimm:	$\texttt{r} \rightarrow \texttt{Plus(r, imm)}$
use:	$\texttt{r} \rightarrow \texttt{Content(ObjAddr a)}$
r2c:	$\texttt{r} \rightarrow \texttt{Const c}$
r2i:	$\texttt{r} \rightarrow \texttt{imm}$
i2c:	$\texttt{imm} \rightarrow \texttt{Const c}$

start symbol ϵ

derivation sequence of rule applications

$<$ add, addimm, r2c $>$



Grammars

nonterminals $N = \{\epsilon, \text{r}, \text{imm}\}$

terminals $\Sigma = \{\texttt{Const, ObjAddr, Content, Plus, Assign}\}$

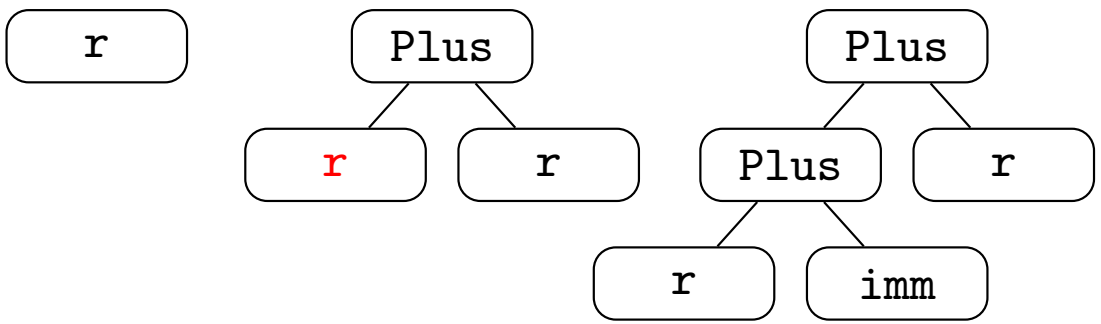
rules (lhs $\rightarrow$ rhs)

def:	$\epsilon \rightarrow \texttt{Assign((ObjAddr a), r)}$
add:	$\texttt{r} \rightarrow \texttt{Plus(r, r)}$
addimm:	$\texttt{r} \rightarrow \texttt{Plus(r, imm)}$
use:	$\texttt{r} \rightarrow \texttt{Content(ObjAddr a)}$
r2c:	$\texttt{r} \rightarrow \texttt{Const c}$
r2i:	$\texttt{r} \rightarrow \texttt{imm}$
i2c:	$\texttt{imm} \rightarrow \texttt{Const c}$

start symbol ϵ

derivation sequence of rule applications

$< \text{add, addimm, r2c} >$



Grammars

nonterminals $N = \{\epsilon, \text{r}, \text{imm}\}$

terminals $\Sigma = \{\texttt{Const, ObjAddr, Content, Plus, Assign}\}$

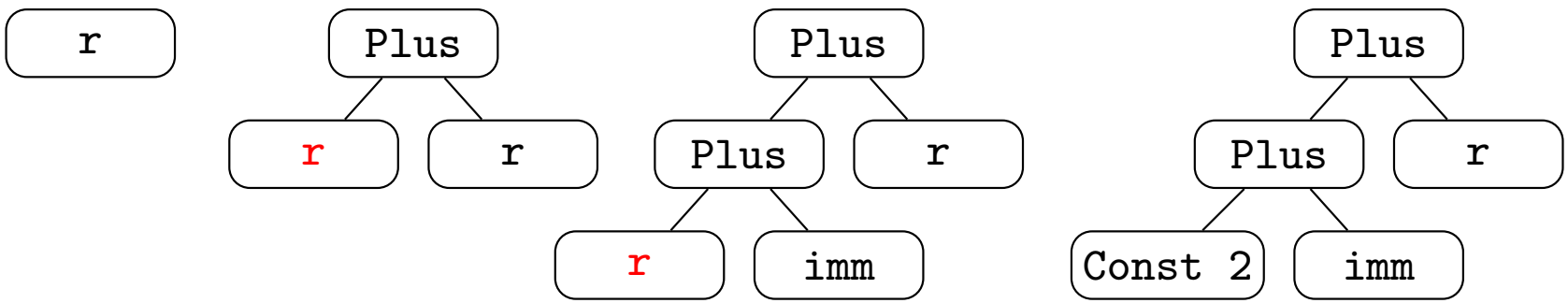
rules (lhs $\rightarrow$ rhs)

def:	$\epsilon \rightarrow \texttt{Assign((ObjAddr a), r)}$
add:	$\texttt{r} \rightarrow \texttt{Plus(r, r)}$
addimm:	$\texttt{r} \rightarrow \texttt{Plus(r, imm)}$
use:	$\texttt{r} \rightarrow \texttt{Content(ObjAddr a)}$
r2c:	$\texttt{r} \rightarrow \texttt{Const c}$
r2i:	$\texttt{r} \rightarrow \texttt{imm}$
i2c:	$\texttt{imm} \rightarrow \texttt{Const c}$

start symbol ϵ

derivation sequence of rule applications

$< \text{add, addimm, r2c} >$



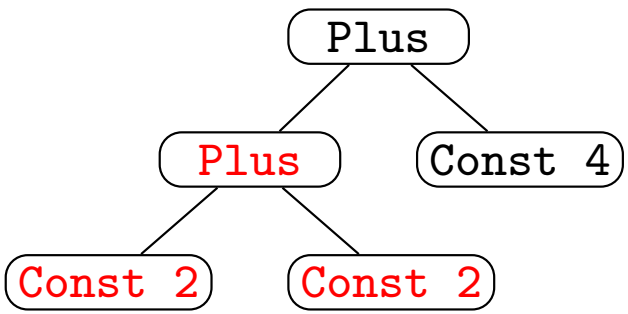
Outline

Symmetry in Derivations

r2c	`r` $\longrightarrow$ `Const c`
i2c	`imm` $\longrightarrow$ `Const c`
add	`r` $\longrightarrow$ `Plus(r, r)`
addimm	`r` $\longrightarrow$ `Plus(r, imm)`

derivation:

add $\Rightarrow$ addimm $\Rightarrow$ r2c $\Rightarrow$ i2c $\Rightarrow$ r2c

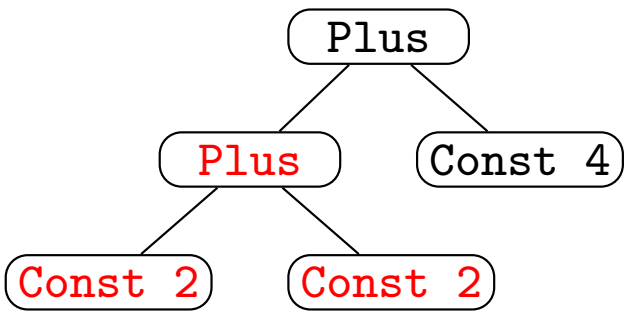


Symmetry in Derivations

r2c	`r` $\longrightarrow$ `Const c`
i2c	`imm` $\longrightarrow$ `Const c`
add	`r` $\longrightarrow$ `Plus(r, r)`
addimm	`r` $\longrightarrow$ `Plus(r, imm)`

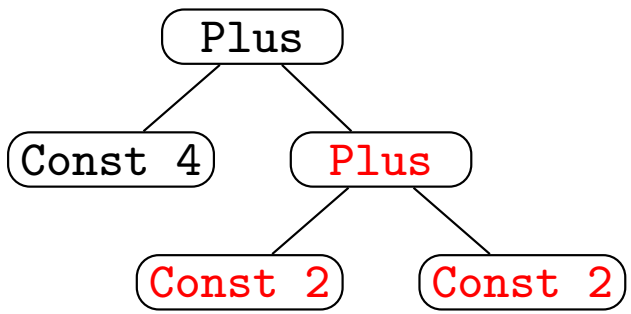
derivation:

add $\Rightarrow$ addimm $\Rightarrow$ r2c $\Rightarrow$ i2c $\Rightarrow$ r2c



derivation:

add $\Rightarrow$ r2c $\Rightarrow$ addimm $\Rightarrow$ r2c $\Rightarrow$ i2c



Principle of Specification Generation

Derivations with Fixed Length

	1	**2**	**3**	**4**	**5**	**6**
1	def	add	add	r2c	r2c	r2c
2	def	add	use	add	r2c	r2c
3	def	add	r2c	add	r2c	r2c
...						
6	def	add	addimm	r2c	i2c	use
7	def	add	addimm	use	i2c	use
...						
30	def	add	use	add	use	use

Principle of Specification Generation

Derivations with Fixed Length

	1	**2**	**3**	**4**	**5**	**6**
1	def	add	add	r2c	r2c	r2c
2	def	add	use	add	r2c	r2c
3	def	add	r2c	add	r2c	r2c
...						
6	def	add	addimm	r2c	i2c	use
7	def	add	addimm	use	i2c	use
...						
30	def	add	use	add	use	use
	P1	P2	P3	P4	P5	P6

derivation length = number of parameters

value of parameter i = rule can be applied in step i

constraints for invalid combinations

CIT Specification Generation

1. Generate compact representation of derivations
2. Calculate value sets
3. Calculate constraints

Generate Compact Representation of Derivations

Recursive Construction:

Base Case derivation length $= 1$
store for terminal rules
lhs nonterminal, rule identifier

Recursion derivation length > 1

Generate Compact Representation of Derivations

Recursive Construction:

Base Case
derivation length $= 1$
store for terminal rules
lhs nonterminal, rule identifier

Recursion
derivation length > 1
check remaining rules
split overall length for number of nonterminals in rule
find smaller derivations
store lhs nonterminal, rule identifier, extended pattern

Generate Compact Representation of Derivations

def:
$\epsilon \rightarrow$`Assign((ObjAddr a),r)`
add:
`r` $\rightarrow$ `Plus(r, r)`
addimm:
`r` $\rightarrow$ `Plus(r, imm)`
use:
`r` $\rightarrow$ `Content(ObjAddr a)`
r2c:
`r` $\rightarrow$ `Const c`
r2i:
`r` $\rightarrow$ `imm`
i2c:
`imm` $\rightarrow$ `Const c`

L	NT	Rule	RHS Pattern
1	r	use	$\emptyset$
		r2c	$\emptyset$
	imm	i2c	$\emptyset$

Generate Compact Representation of Derivations

def:
$\epsilon \rightarrow$`Assign((ObjAddr a),r)`
add:
`r` $\rightarrow$ `Plus(r, r)`
addimm:
`r` $\rightarrow$ `Plus(r, imm)`
use:
`r` $\rightarrow$ `Content(ObjAddr a)`
r2c:
`r` $\rightarrow$ `Const c`
r2i:
`r` $\rightarrow$ `imm`
i2c:
`imm` $\rightarrow$ `Const c`

L	NT	Rule	RHS Pattern
1	r	use	$\emptyset$
		r2c	$\emptyset$
	imm	i2c	$\emptyset$
2	r	r2i	$<$ (1, imm) $>$
	ϵ	def	$<$ (1, reg) $>$
3	r	addimm	$<$ (1, reg), (1, imm) $>$
		add	$<$ (1, r), (1, r) $>$
	ϵ	def	$<$ (2,r) $>$

Generate Compact Representation of Derivations

def:
$\epsilon \rightarrow$`Assign((ObjAddr a),r)`
add:
`r` $\rightarrow$ `Plus(r, r)`
addimm:
`r` $\rightarrow$ `Plus(r, imm)`
use:
`r` $\rightarrow$ `Content(ObjAddr a)`
r2c:
`r` $\rightarrow$ `Const c`
r2i:
`r` $\rightarrow$ `imm`
i2c:
`imm` $\rightarrow$ `Const c`

L	NT	Rule	RHS Pattern
1	r	use	$\emptyset$
		r2c	$\emptyset$
	imm	i2c	$\emptyset$
2	r	r2i	$<$ (1, imm) $>$
	ϵ	def	$<$ (1, reg) $>$
3	r	addimm	$<$ (1, reg), (1, imm) $>$
		add	$<$ (1, r), (1, r) $>$
	ϵ	def	$<$ (2,r) $>$
4		add	$<$ (1, r), (2, r) $>$
			$<$ (2, r), (1, r) $>$

Generate Value Sets

Recursive Construction:

Base Case derivation length $= 1$
collect rules for length 1

Recursion derivation length > 1
iterate over rules with considered nonterminal
extend parameter with rules
for each rule iterate over alternatives
descend for each nonterminal

Generate Compact Representation of Derivations

L	NT	Rule	RHS Pattern
1	r	use	$\emptyset$
		r2c	$\emptyset$
	imm	i2c	$\emptyset$
2	r	r2i	$<$ (1, imm) $>$
	ϵ	def	$<$ (1, r) $>$
3	r	addimm	$<$ (1, r), (1, imm) $>$
		add	$<$ (1, r), (1, r) $>$
	ϵ	def	$<$ (2,reg) $>$
4	r	addimm	$<$ (2, r), (1, imm) $>$
		add	$<$ (1, r), (2, r) $>$
			$<$ (2, r), (1, r) $>$
	ϵ	def	$<$ (3,r) $>$

P1
P2
P3
P4

Generate Compact Representation of Derivations

L	NT	Rule	RHS Pattern
1	r	use	$\emptyset$
		r2c	$\emptyset$
	imm	i2c	$\emptyset$
2	r	r2i	$<$ (1, imm) $>$
	ϵ	def	$<$ (1, r) $>$
3	r	addimm	$<$ (1, r), (1, imm) $>$
		add	$<$ (1, r), (1, r) $>$
	ϵ	def	$<$ (2,reg) $>$
4	r	addimm	$<$ (2, r), (1, imm) $>$
		add	$<$ (1, r), (2, r) $>$
			$<$ (2, r), (1, r) $>$
	ϵ	def	$<$ (3,r) $>$

P1 def
P2
P3
P4

Generate Compact Representation of Derivations

L	NT	Rule	RHS Pattern
1	r	use	$\emptyset$
		r2c	$\emptyset$
	imm	i2c	$\emptyset$
2	r	r2i	$<$ (1, imm) $>$
	ϵ	def	$<$ (1, r) $>$
3	r	addimm	$<$ (1, r), (1, imm) $>$
		add	$<$ (1, r), (1, r) $>$
	ϵ	def	$<$ (2,reg) $>$
4	r	addimm	$<$ (2, r), (1, imm) $>$
		add	$<$ (1, r), (2, r) $>$
			$<$ (2, r), (1, r) $>$
	ϵ	def	$<$ (3,r) $>$

P1	def
P2	addimm
P3	
P4	

Generate Compact Representation of Derivations

L	NT	Rule	RHS Pattern
1	r	use	$\emptyset$
		r2c	$\emptyset$
	imm	i2c	$\emptyset$
2	r	r2i	$<$ (1, imm) $>$
	ϵ	def	$<$ (1, r) $>$
3	r	addimm	$<$ (1, r), (1, imm) $>$
		add	$<$ (1, r), (1, r) $>$
	ϵ	def	$<$ (2,reg) $>$
4	r	addimm	$<$ (2, r), (1, imm) $>$
		add	$<$ (1, r), (2, r) $>$
			$<$ (2, r), (1, r) $>$
	ϵ	def	$<$ (3,r) $>$

P1	def
P2	add
P3	use, r2c
P4	

Generate Compact Representation of Derivations

L	NT	Rule	RHS Pattern
1	r	use	$\emptyset$
		r2c	$\emptyset$
	imm	i2c	$\emptyset$
2	r	r2i	$<$ (1, imm) $>$
	ϵ	def	$<$ (1, r) $>$
3	r	addimm	$<$ (1, r), (1, imm) $>$
		add	$<$ (1, r), (1, r) $>$
	ϵ	def	$<$ (2,reg) $>$
4	r	addimm	$<$ (2, r), (1, imm) $>$
		add	$<$ (1, r), (2, r) $>$
			$<$ (2, r), (1, r) $>$
	ϵ	def	$<$ (3,r) $>$

P1	def
P2	addimm
P3	use, r2c
P4	i2c

Generate Constraints

Recursive Construction:

Base Case derivation length $= 1$
disjunction of all rules for length 1

Recursion derivation length > 1
iterate over rules with considered nonterminal
  construct disjunction
for each rule iterate over alternatives
  construct disjunction
for each alternative
  construct conjunction

Generate Compact Representation of Derivations

L	NT	Rule	RHS Pattern
1	r	use	$\emptyset$
		r2c	$\emptyset$
	imm	i2c	$\emptyset$
2	r	r2i	$<$ (1, imm) $>$
	ϵ	def	$<$ (1, r) $>$
3	r	addimm	$<$ (1, r), (1, imm) $>$
		add	$<$ (1, r), (1, r) $>$
	ϵ	def	$<$ (2,reg) $>$
4	r	addimm	$<$ (2, r), (1, imm) $>$
		add	$<$ (1, r), (2, r) $>$
			$<$ (2, r), (1, r) $>$
	ϵ	def	$<$ (3,r) $>$

Generate Compact Representation of Derivations

L	NT	Rule	RHS Pattern
1	r	use	$\emptyset$
		r2c	$\emptyset$
	imm	i2c	$\emptyset$
2	r	r2i	$<$ (1, imm) $>$
	ϵ	def	$<$ (1, r) $>$
3	r	addimm	$<$ (1, r), (1, imm) $>$
		add	$<$ (1, r), (1, r) $>$
	ϵ	def	$<$ (2,reg) $>$
4	r	addimm	$<$ (2, r), (1, imm) $>$
		add	$<$ (1, r), (2, r) $>$
			$<$ (2, r), (1, r) $>$
	ϵ	def	$<$ (3,r) $>$

def $\longrightarrow f$

Generate Compact Representation of Derivations

L	NT	Rule	RHS Pattern
1	r	use	$\emptyset$
		r2c	$\emptyset$
	imm	i2c	$\emptyset$
2	r	r2i	$<$ (1, imm) $>$
	ϵ	def	$<$ (1, r) $>$
3	r	addimm	$<$ (1, r), (1, imm) $>$
		add	$<$ (1, r), (1, r) $>$
	ϵ	def	$<$ (2,reg) $>$
4	r	addimm	$<$ (2, r), (1, imm) $>$
		add	$<$ (1, r), (2, r) $>$
			$<$ (2, r), (1, r) $>$
	ϵ	def	$<$ (3,r) $>$

def $\longrightarrow$
(addimm $\wedge$ f)

Generate Compact Representation of Derivations

L	NT	Rule	RHS Pattern
1	r	use	$\emptyset$
		r2c	$\emptyset$
	imm	i2c	$\emptyset$
2	r	r2i	$<$ (1, imm) $>$
	ϵ	def	$<$ (1, r) $>$
3	r	addimm	$<$ (1, r), (1, imm) $>$
		add	$<$ (1, r), (1, r) $>$
	ϵ	def	$<$ (2,reg) $>$
4	r	addimm	$<$ (2, r), (1, imm) $>$
		add	$<$ (1, r), (2, r) $>$
			$<$ (2, r), (1, r) $>$
	ϵ	def	$<$ (3,r) $>$

def $\longrightarrow$
(addimm $\wedge$
((use $\vee$ r2c) $\wedge$ f))

Generate Compact Representation of Derivations

L	NT	Rule	RHS Pattern
1	r	use	$\emptyset$
		r2c	$\emptyset$
	imm	i2c	$\emptyset$
2	r	r2i	$<$ (1, imm) $>$
	ϵ	def	$<$ (1, r) $>$
3	r	addimm	$<$ (1, r), (1, imm) $>$
		add	$<$ (1, r), (1, r) $>$
	ϵ	def	$<$ (2,reg) $>$
4	r	addimm	$<$ (2, r), (1, imm) $>$
		add	$<$ (1, r), (2, r) $>$
			$<$ (2, r), (1, r) $>$
	ϵ	def	$<$ (3,r) $>$

def $\longrightarrow$
(addimm $\wedge$
((use $\vee$ r2c) $\wedge$ i2c))

Outline

Size of Parameters

		P_1	P_2	P_3	P_4	P_5	P_6
G1	Spec3	5	17	9			
	Spec4	10	38	26	13		
	Spec5	10	43	38	26	13	
	Spec6	12	55	47	38	26	13
G2	Spec3	5	21	13			
	Spec4	10	44	30	20		
	Spec5	10	53	47	32	20	
	Spec6	12	88	67	54	32	20

G1
86 rules
12 initial rules
G2
110 rules
12 initial rules

➠ generation within seconds

Size of Test Sets

L		G1 5 (1)	G2 6 (2)	G3 6 (1)	G4 7 (2)
6	Spec	$1\ 2^3\ 4^2$	$2^4\ 4^2$	$1\ 2^2\ 3\ 5^2$	$2^3\ 3\ 5^2$
	Derivations	64	128	144	288
	TCases(ACTS)	16	25	13	22
8	Spec	$1\ 2^3\ 4^4$	$2^4\ 4^4$	$1\ 2^2\ 3\ 5^4$	$2^3\ 3\ 5^4$
	Derivations	640	1280	2160	4320
	TCases(ACTS)	22	32	17	28
10	Spec	$1\ 2^3\ 4^6$	$2^4\ 4^6$	$1\ 2^2\ 3\ 5^6$	$2^3\ 3\ 5^6$
	Derivations	7168	14336	36288	-
	TCases(ACTS)	32	38	24	30

Size of Test Sets

L		G1 5 (1)	G2 6 (2)	G3 6 (1)	G4 7 (2)
6	Spec	$1\ 2^3\ 4^2$	$2^4\ 4^2$	$1\ 2^2\ 3\ 5^2$	$2^3\ 3\ 5^2$
	Derivations	64	128	144	288
	TCases(ACTS)	16	25	13	22
8	Spec	$1\ 2^3\ 4^4$	$2^4\ 4^4$	$1\ 2^2\ 3\ 5^4$	$2^3\ 3\ 5^4$
	Derivations	640	1280	2160	4320
	TCases(ACTS)	22	32	17	28
10	Spec	$1\ 2^3\ 4^6$	$2^4\ 4^6$	$1\ 2^2\ 3\ 5^6$	$2^3\ 3\ 5^6$
	Derivations	7168	14336	36288	-
	TCases(ACTS)	32	38	24	30

- significant reduction
- generation failed for case study grammars

Conclusion and Future Work

Conclusion

- new approach for grammar-based testing
  - criterion guarantees t-wise rule coverage
  - test selection based on combinatorial interaction testing
  - CIT specification automatically generated
- very encouraging results regarding test set size

Conclusion and Future Work

Conclusion

- new approach for grammar-based testing
  - criterion guarantees t-wise rule coverage
  - test selection based on combinatorial interaction testing
  - CIT specification automatically generated
- very encouraging results regarding test set size

Future Work

- enhancement of constraint generation
- combination with alternative approach for grammar-based testing
- comprehensive case study for compiler back ends