Concordia University
Engineering and Computer Science

Coverage Driven Test Generation and Consistency Algorithm

Jomu George

Dr Otmane Ait Mohamed

Hardware Verification Group (HVG)
Department of Electrical and Computer Engineering
Concordia University, Canada

AGENDA

- Coverage Driven Test Generation(CDTG)
- Motivation
- Related work
- Why Generalized Arc Consistency Algorithm
- Intuitive idea of Proposed Algorithm
- Experimental Results
- Conclusion

AGENDA

- Coverage Driven Test Generation(CDTG)
- Motivation
- Related work
- Why Generalized Arc Consistency Algorithm
- Intuitive idea of Proposed Algorithm
- Experimental Results
- Conclusion

AGENDA

- Coverage Driven Test Generation(CDTG)
- Motivation
- Related work
- Why Generalized Arc Consistency Algorithm
- Intuitive idea of Proposed Algorithm
- Experimental Results
- Conclusion

AGENDA

- Coverage Driven Test Generation(CDTG)
- Motivation
- Related work
- Why Generalized Arc Consistency Algorithm
- Intuitive idea of Proposed Algorithm
- Experimental Results
- Conclusion

AGENDA

- Coverage Driven Test Generation(CDTG)
- Motivation
- Related work
- Why Generalized Arc Consistency Algorithm
- Intuitive idea of Proposed Algorithm
- Experimental Results
- Conclusion

AGENDA

- Coverage Driven Test Generation(CDTG)
- Motivation
- Related work
- Why Generalized Arc Consistency Algorithm
- Intuitive idea of Proposed Algorithm
- Experimental Results
- Conclusion

AGENDA

- Coverage Driven Test Generation(CDTG)
- Motivation
- Related work
- Why Generalized Arc Consistency Algorithm
- Intuitive idea of Proposed Algorithm
- Experimental Results
- Conclusion

AGENDA

- Coverage Driven Test Generation(CDTG)
- Motivation
- Related work
- Why Generalized Arc Consistency Algorithm
- Intuitive idea of Proposed Algorithm
- Experimental Results
- Conclusion

Coverage Driven Test Generation

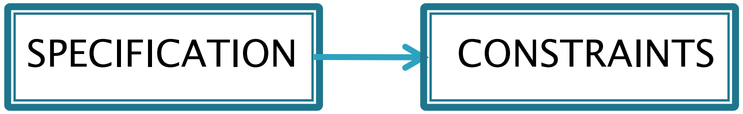
- Coverage driven test generation (CDTG) is a technique in which coverage analysis report is used to direct the next test generation.

Coverage Driven Test Generation

SPECIFICATION

- Coverage driven test generation (CDTG) is a technique in which coverage analysis report is used to direct the next test generation.

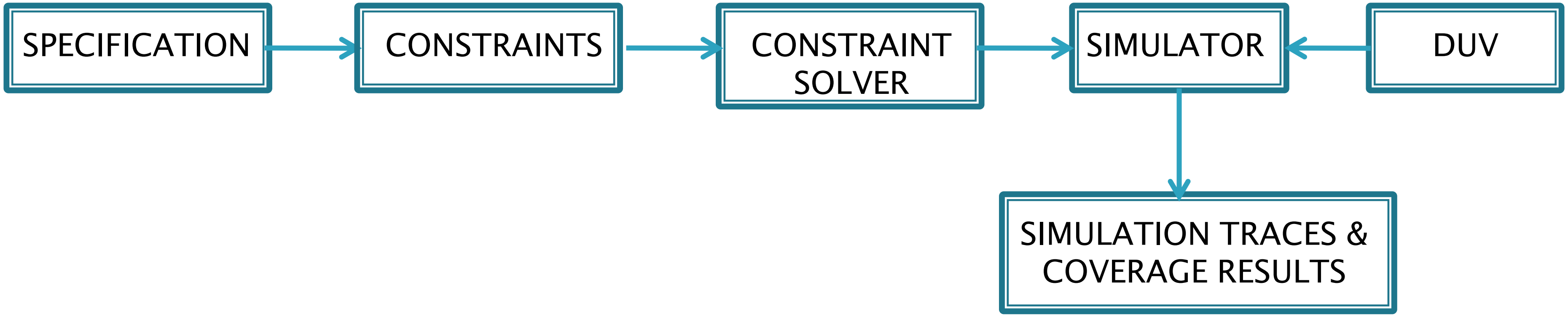
Coverage Driven Test Generation



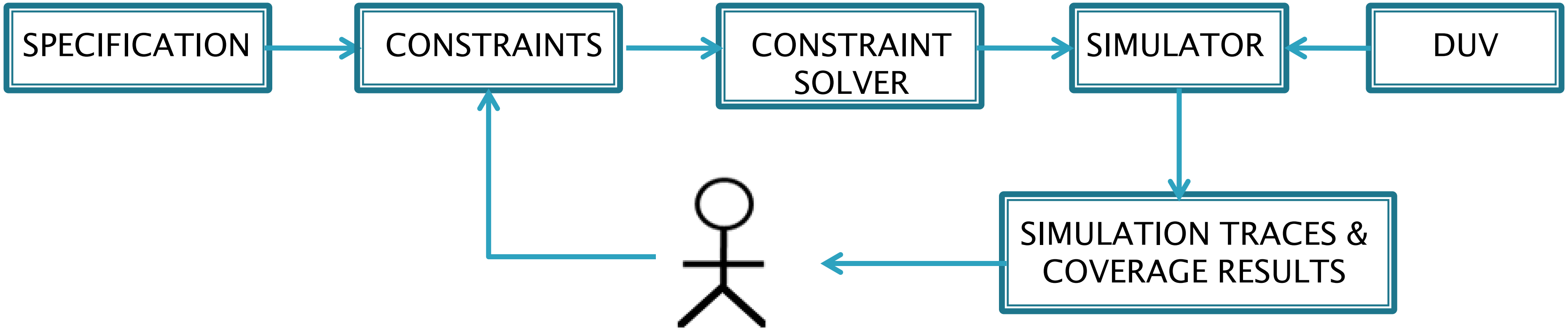
Coverage Driven Test Generation

Coverage Driven Test Generation

Coverage Driven Test Generation

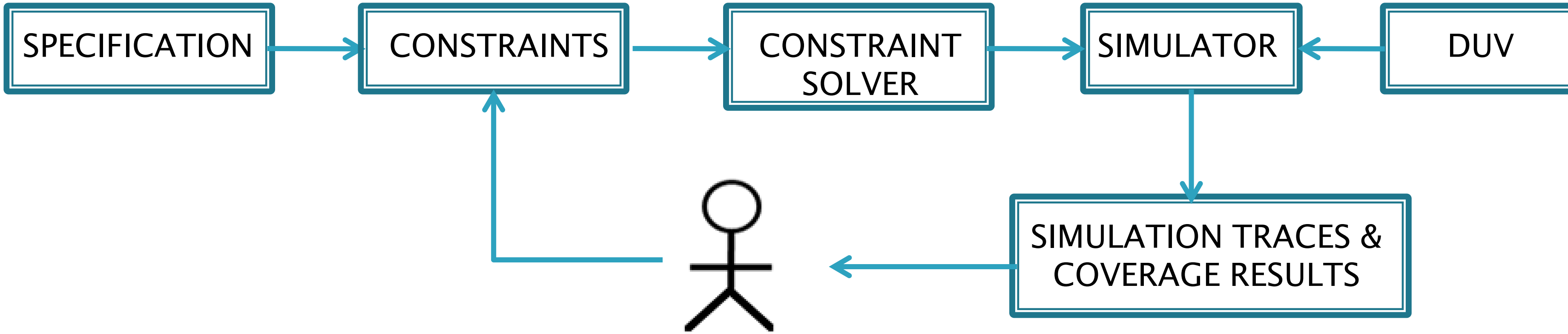


Coverage Driven Test Generation



- Coverage driven test generation (CDTG) is a technique in which coverage analysis report is used to direct the next test generation.

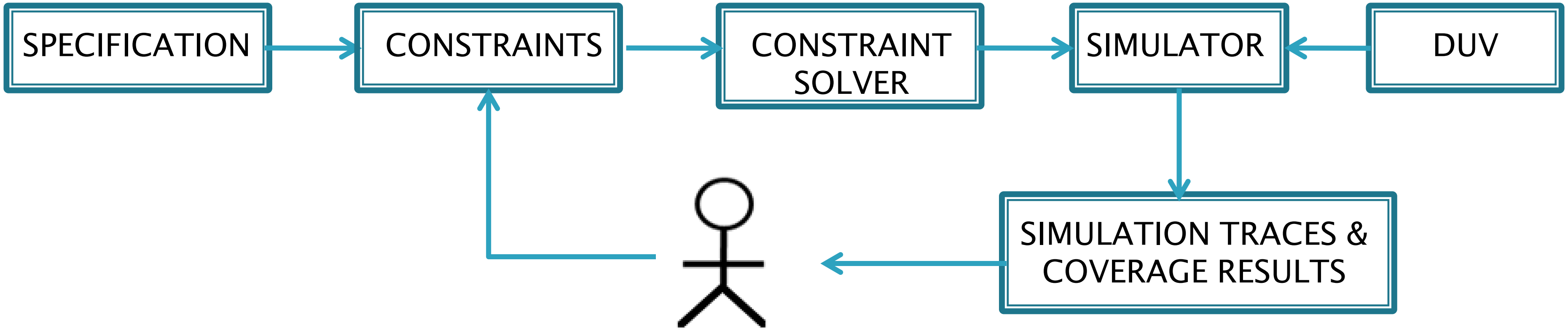
Coverage Driven Test Generation



- Coverage driven test generation (CDTG) is a technique in which coverage analysis report is used to direct the next test generation.

- There are two benefits for CDTG.
  - Unobserved scenarios will be generated.
  - Certain scenarios can be more easily tested multiple times with different input parameters.

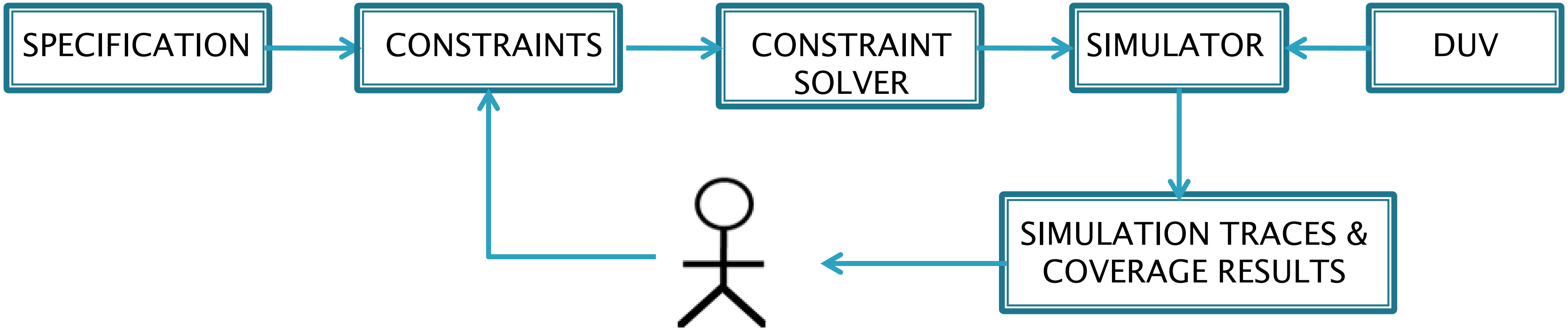
Coverage Driven Test Generation



- Coverage driven test generation (CDTG) is a technique in which coverage analysis report is used to direct the next test generation.

- There are two benefits for CDTG.
  - Unobserved scenarios will be generated.
  - Certain scenarios can be more easily tested multiple times with different input parameters.

Coverage Driven Test Generation



- Coverage driven test generation (CDTG) is a technique in which coverage analysis report is used to direct the next test generation.

- There are two benefits for CDTG.
  - Unobserved scenarios will be generated.
  - Certain scenarios can be more easily tested multiple times with different input parameters.

Motivation

- Laurent Fournier, Yaron Arbetman and Moshe Levinger ,2007:
- Probability that a CDTG tool (Genesys) will generate a sequence that covers a particular combination is very low
  Consider a floating point unit:
  - 2 input operands,
  - 20 major FP instruction types: normalized, denormalized, zero, infinity, ……
  - 4 floating point instructions : addition, subtraction, division and multiplication
  - ➔ based on random generation

- Yingpan Wu,Lixin Yu, Wei Zhuang and Jianyong Wang ,2009
  - Verification of Data hazard for a microprocessor takes about 6 days
  - Verification of Control hazard for a microprocessor takes about 9 days
  - ➔ constrained random generation

Motivation

- Laurent Fournier, Yaron Arbetman and Moshe Levinger ,2007:
- Probability that a CDTG tool (Genesys) will generate a sequence that covers a particular combination is very low Consider a floating point unit:
  - 2 input operands,
  - 20 major FP instruction types: normalized, denormalized, zero, infinity, ……
  - 4 floating point instructions : addition, subtraction, division and multiplication
  - ➔ based on random generation
- Yingpan Wu,Lixin Yu, Wei Zhuang and Jianyong Wang ,2009
  - Verification of Data hazard for a microprocessor takes about 6 days
  - Verification of Control hazard for a microprocessor takes about 9 days
  - ➔ constrained random generation

Motivation

- Laurent Fournier, Yaron Arbetman and Moshe Levinger ,2007:
- Probability that a CDTG tool (Genesys) will generate a sequence that covers a particular combination is very low Consider a floating point unit:
  - 2 input operands,
  - 20 major FP instruction types: normalized, denormalized, zero, infinity, ……
  - 4 floating point instructions : addition, subtraction, division and multiplication
  - ➔ based on random generation

- Yingpan Wu,Lixin Yu, Wei Zhuang and Jianyong Wang ,2009
  - Verification of Data hazard for a microprocessor takes about 6 days
  - Verification of Control hazard for a microprocessor takes about 9 days
  - ➔ constrained random generation

Motivation

- Laurent Fournier, Yaron Arbetman and Moshe Levinger ,2007:
- Probability that a CDTG tool (Genesys) will generate a sequence that covers a particular combination is very low Consider a floating point unit:
  - 2 input operands,
  - 20 major FP instruction types: normalized, denormalized, zero, infinity, ……
  - 4 floating point instructions : addition, subtraction, division and multiplication
  - ➔ based on random generation

- Yingpan Wu,Lixin Yu, Wei Zhuang and Jianyong Wang ,2009
  - Verification of Data hazard for a microprocessor takes about 6 days
  - Verification of Control hazard for a microprocessor takes about 9 days
  - ➔ constrained random generation

Motivation

- Laurent Fournier, Yaron Arbetman and Moshe Levinger ,2007:
- Probability that a CDTG tool (Genesys) will generate a sequence that covers a particular combination is very low for Consider a floating point unit:
  - 2 input operands,
  - 20 major FP instruction types: normalized, denormalized, zero, infinity, ……
  - 4 floating point instructions : addition, subtraction, division and multiplication
  - ➔ based on random generation

- Yingpan Wu,Lixin Yu, Wei Zhuang and Jianyong Wang ,2009
  - Verification of Data hazard for a microprocessor takes about 6 days
  - Verification of Control hazard for a microprocessor takes about 9 days
  - ➔ constrained random generation

Motivation

- Laurent Fournier, Yaron Arbetman and Moshe Levinger ,2007:
- Probability that a CDTG tool (Genesys) will generate a sequence that covers a particular combination is very low for Consider a floating point unit:
  - 2 input operands,
  - 20 major FP instruction types: normalized, denormalized, zero, infinity, ……
  - 4 floating point instructions : addition, subtraction, division and multiplication
  - ➔ based on random generation

- Yingpan Wu,Lixin Yu, Wei Zhuang and Jianyong Wang ,2009
  - Verification of Data hazard for a microprocessor takes about 6 days
  - Verification of Control hazard for a microprocessor takes about 9 days
  - ➔ constrained random generation

Constraint Solver

- The CDTG must have two parts:
  - Constraint models or language
  - Constraint solver engine
- CDTG has the following disadvantages:
  - Solving constraints requires a lot of time.
  - The memory required is very large for constraints with large variable.
- Solvers of CSP are different from CDTG:
  - Multiple different solutions for same problem
  - Variables have huge domains
  - Non Uniformity

Constraint Solver

- The CDTG must have two parts:
  - Constraint models or language
  - Constraint solver engine
- Solvers of CDTG has the following disadvantages:
  - Solving constraints requires a lot of time.
  - The memory required is very large for constraints with large variable.
- Solvers of CSP are different from CDTG:
  - Multiple different solutions for same problem
  - Variables have huge domains
  - Non Uniformity

Constraint Solver

- The CDTG must have two parts:
  - Constraint models or language
  - Constraint solver engine
- Solvers of CDTG has the following disadvantages:
  - Solving constraints requires a lot of time.
  - The memory required is very large for constraints with large variable.
- Solvers of CSP are different from CDTG:
  - Multiple different solutions for same problem
  - Variables have huge domains
  - Non Uniformity

Constraint Solver

- The CDTG must have two parts:
  - Constraint models or language
  - Constraint solver engine
- Solvers of CDTG has the following disadvantages:
  - Solving constraints requires a lot of time.
  - The memory required is very large for constraints with large variable.
- Solvers of CSP are different from CDTG:
  - Multiple different solutions for same problem
  - Variables have huge domains
  - Non Uniformity

Constraint Solver

- The CDTG must have two parts:
  - Constraint models or language
  - Constraint solver engine
- Solvers of CDTG has the following disadvantages:
  - Solving constraints requires a lot of time.
  - The memory required is very large for constraints with large variable.
- Solvers of CSP are different from CDTG:
  - Multiple different solutions for same problem
  - Variables have huge domains
  - Non Uniformity

Constraint Solver

- The CDTG must have two parts:
  - Constraint models or language
  - Constraint solver engine
- CDTG has the following disadvantages:
  - Solving constraints requires a lot of time.
  - The memory required is very large for constraints with large variable.
- Solvers of CSP are different from CDTG:
  - Multiple different solutions for same problem
  - Variables have huge domains
  - Non Uniformity

Problem Solution

- The efficiency of the solver can be improved by reducing the search space.
- Search space can be reduced by removing inconsistent values.
- **Idea**: To prune the variable domains as much as possible before selecting values from them.
- Consistency Search Algorithms

Problem Solution

- The efficiency of the solver can be improved by reducing the search space.
- Search space can be reduced by removing inconsistent values.
- **Idea**: To prune the variable domains as much as possible before selecting values from them.
- Consistency Search Algorithms

Problem Solution

- The efficiency of the solver can be improved by reducing the search space.
- Search space can be reduced by removing inconsistent values.
- **Idea**: To prune the variable domains as much as possible before selecting values from them.
- Consistency Search Algorithms

Problem Solution

- The efficiency of the solver can be improved by reducing the search space.
- Search space can be reduced by removing inconsistent values.
- **Idea**: To prune the variable domains as much as possible before selecting values from them.
- Consistency Search Algorithms

Related Work

- Coarse grained algorithms
  - The removal of a value from the domain of a variable will be propagated to all other variables in the problem
  - AC-1, AC-3, AC2000, AC2001, AC2001-OP, AC3.1, AC3-OP, AC3d

- Fine grained consistency algorithms
  - The removal of a value from the domain of a variable 'X' will affect only other variables which are related to the variable 'X'.
  - AC-4, AC4-OP, AC-5, AC-6
  - AC-7 for n-arity constraints in GAC
  - GAC-scheme on conjunctions of constraints.

Related Work

- Coarse grained algorithms
  - The removal of a value from the domain of a variable will be propagated to all other variables in the problem
  - AC-1, AC-3, AC2000, AC2001, AC2001-OP, AC3.1, AC3-OP, AC3d

- Fine grained consistency algorithms
  - The removal of a value from the domain of a variable 'X' will affect only other variables which are related to the variable 'X'.
  - AC-4, AC4-OP, AC-5, AC-6
  - AC-7 for n-arity constraints is GAC
  - GAC-scheme on conjunctions of constraints.

Related Work

- Coarse grained algorithms
  - The removal of a value from the domain of a variable will be propagated to all other variables in the problem
  - AC-1, AC-3, AC2000, AC2001, AC2001-OP, AC3.1, AC3-OP, AC3d
- Fine grained consistency algorithms
  - The removal of a value from the domain of a variable 'X' will affect only other variables which are related to the variable 'X'.
  - AC-4, AC4-OP, AC-5, AC-6
  - AC-7 for n-arity constraints is GAC
  - GAC-scheme on conjunctions of constraints.

Related Work

- Coarse grained algorithms
  - The removal of a value from the domain of a variable will be propagated to all other variables in the problem
  - AC-1, AC-3, AC2000, AC2001, AC2001-OP, AC3.1, AC3-OP, AC3d

- Fine grained consistency algorithms
  - The removal of a value from the domain of a variable 'X' will affect only other variables which are related to the variable 'X'.
  - AC-4, AC4-OP, AC-5, AC-6
  - AC-7 for n-arity constraints is GAC
  - GAC-scheme on conjunctions of constraints.

Why GACCC

- The constraints used in CDTG can have more than two variables and GAC-scheme can handle constraint of n-arity.
- We need to eliminate as much invalid domain values as possible. This can be done by conjunction of constraints.
- GAC scheme do not require any specific data structure.
- The constraints used in CDTG are not of a fixed type and GAC-scheme can be used with any type of constraints.

Why GACCC

- The constraints used in CDTG can have more than two variables and GAC-scheme can handle constraint of n-arity.
- We need to eliminate as much invalid domain values as possible. This can be done by conjunction of constraints.
- GAC scheme do not require any specific data structure.
- The constraints used in CDTG are not of a fixed type and GAC-scheme can be used with any type of constraints.

Why GACCC

- The constraints used in CDTG can have more than two variables and GAC-scheme can handle constraint of n-arity.
- We need to eliminate as much invalid domain values as possible. This can be done by conjunction of constraints.
- GAC scheme do not require any specific data structure.
- The constraints used in CDTG are not of a fixed type and GAC-scheme can be used with any type of constraints.

Why GACCC

- The constraints used in CDTG can have more than two variables and GAC-scheme can handle constraint of n-arity.
- We need to eliminate as much invalid domain values as possible. This can be done by conjunction of constraints.
- GAC scheme do not require any specific data structure.
- The constraints used in CDTG are not of a fixed type and GAC-scheme can be used with any type of constraints.

GACCC

Set of variables X= {m, n, o, p, q}.
Domain of the variables, D(m)={1, 2}, D(n)={2,3} , D(o)={1, 2} , D(p)={1, 3} , D(q)={2, 3}.
C1: m+n+o+p=10 and C2: n+o+q=9

GACCC

Set of variables X= {m, n, o, p, q}.
Domain of the variables, D(m)={1, 2}, D(n)={2,3} , D(o)={1, 2} , D(p)={1, 3} , D(q)={2, 3}.
C1: m+n+o+p=10 and C2: n+o+q=9

M	N	O	P	Q
1	2	1	1	2
1	2	1	1	3
1	2	1	3	2
1	2	1	3	3
1	2	2	1	2
1	2	2	1	3
1	2	2	3	2
1	2	2	3	3
1	3	1	1	2
1	3	1	1	3
1	3	1	3	2
1	3	1	3	3
1	3	2	1	2
1	3	2	1	3
1	3	2	3	2
1	3	2	3	3

GACCC

Set of variables X= {m, n, o, p, q}.
Domain of the variables, D(m)={1, 2}, D(n)={2,3} , D(o)={1, 2} , D(p)={1, 3} , D(q)={2, 3}.
C1: m+n+o+p=10 and C2: n+o+q=9

16 tuples m=1 inconsistent

M	N	O	P	Q
1	2	1	1	2
1	2	1	1	3
1	2	1	3	2
1	2	1	3	3
1	2	2	1	2
1	2	2	1	3
1	2	2	3	2
1	2	2	3	3
1	3	1	1	2
1	3	1	1	3
1	3	1	3	2
1	3	1	3	3
1	3	2	1	2
1	3	2	1	3
1	3	2	3	2
1	3	2	3	3

Intuitive Idea of GACCC-op

Set of variables X= {m, n, o, p, q}.
Domain of the variables, D(m)={1, 2}, D(n)={2,3} , D(o)={1, 2} , D(p)={1, 3} , D(q)={2, 3}.
C1: m+n+o+p=10 and C2: n+o+q=9

M	N	O	P	Q
1	2	1	1	2
1	2	1	1	3
1	2	1	3	2
1	2	1	3	3
1	2	2	1	2
1	2	2	1	3
1	2	2	3	2
1	2	2	3	3
1	3	1	1	2
1	3	1	1	3
1	3	1	3	2
1	3	1	3	3
1	3	2	1	2
1	3	2	1	3
1	3	2	3	2
1	3	2	3	3

Intuitive Idea of GACCC-op

Set of variables X= {m, n, o, p, q}.
Domain of the variables, D(m)={1, 2}, D(n)={2,3} , D(o)={1, 2} , D(p)={1, 3} , D(q)={2, 3}.
C1: m+n+o+p=10 and C2: n+o+q=9

8 tuples m=1 inconsistent

M	N	O	P	Q
1	2	1	1	2
1	2	1	1	3
1	2	1	3	2
1	2	1	3	3
1	2	2	1	2
1	2	2	1	3
1	2	2	3	2
1	2	2	3	3
1	3	1	1	2
1	3	1	1	3
1	3	1	3	2
1	3	1	3	3
1	3	2	1	2
1	3	2	1	3
1	3	2	3	2
1	3	2	3	3

Intuitive Idea of GACCC-op

Set of variables X= {m, n, o, p, q}.
Domain of the variables, D(m)={1, 2}, D(n)={2,3} , D(o)={1, 2} , D(p)={1, 3} , D(q)={2, 3}.
C3: m+n+o+p=6 and C4: n+o+q=7

M	N	O	P	Q
1	2	1	1	2
1	2	1	1	3
1	2	1	3	2
1	2	1	3	3
1	2	2	1	2
1	2	2	1	3
1	2	2	3	2
1	2	2	3	3
1	3	1	1	2
1	3	1	1	3
1	3	1	3	2
1	3	1	3	3
1	3	2	1	2
1	3	2	1	3
1	3	2	3	2
1	3	2	3	3

Intuitive Idea of GACCC-op

Set of variables X= {m, n, o, p, q}.
Domain of the variables, D(m)={1, 2}, D(n)={2,3} , D(o)={1, 2} , D(p)={1, 3} , D(q)={2, 3}.
C3: m+n+o+p=6 and C4: n+o+q=7

M	N	O	P	Q
1	2	1	1	2
1	2	1	1	3
1	2	1	3	2
1	2	1	3	3
1	2	2	1	2
1	2	2	1	3
1	2	2	3	2
1	2	2	3	3
1	3	1	1	2
1	3	1	1	3
1	3	1	3	2
1	3	1	3	3
1	3	2	1	2
1	3	2	1	3
1	3	2	3	2
1	3	2	3	3

Intuitive Idea of GACCC-op

Set of variables X= {m, n, o, p, q}.
Domain of the variables, D(m)={1, 2}, D(n)={2,3} , D(o)={1, 2} , D(p)={1, 3} , D(q)={2, 3}.
C3: m+n+o+p=6 and C4: n+o+q=7

M	N	O	P	Q
1	2	1	1	2
1	2	1	1	3
1	2	1	3	2
1	2	1	3	3
1	2	2	1	2
1	2	2	1	3
1	2	2	3	2
1	2	2	3	3
1	3	1	1	2
1	3	1	1	3
1	3	1	3	2
1	3	1	3	3
1	3	2	1	2
1	3	2	1	3
1	3	2	3	2
1	3	2	3	3

Intuitive Idea of GACCC-op

Set of variables X= {m, n, o, p, q}.
Domain of the variables, D(m)={1, 2}, D(n)={2,3} , D(o)={1, 2} , D(p)={1, 3} , D(q)={2, 3}.
C3: m+n+o+p=6 and C4: n+o+q=7

M	N	O	P	Q
1	2	1	1	2
1	2	1	1	3
1	2	1	3	2
1	2	1	3	3
1	2	2	1	2
1	2	2	1	3
1	2	2	3	2
1	2	2	3	3
1	3	1	1	2
1	3	1	1	3
1	3	1	3	2
1	3	1	3	3
1	3	2	1	2
1	3	2	1	3
1	3	2	3	2
1	3	2	3	3

GACCC & GACCC-op

- In GACCC the support list is made by using some existing variable order scheme.
- In GACCC-op we propose a new variable ordering scheme.
  - The variable, which is present in the constraint with the lowest arity.
  - Has the largest number of domain values.

- In GACCC during consistency search of a domain value of a variable, the tuples generated will contain all the variable in the conjunction set.
- In GACCC-op the consistency search for a variable x
- Will begin with tuples which contain only variables from the smallest constraint(Cs)
- Cs should contain the variable x.

GACCC & GACCC-op

- In GACCC the support list is made by using some existing variable order scheme.
- In GACCC-op we propose a new variable ordering scheme.
  - The variable, which is present in the constraint with the lowest arity.
  - Has the largest number of domain values.

- In GACCC during consistency search of a domain value of a variable, the tuples generated will contain all the variable in the conjunction set.
- In GACCC-op the consistency search for a variable x
- Will begin with tuples which contain only variables from the smallest constraint(Cs)
- Cs should contain the variable x.

GACCC & GACCC-op

- In GACCC the support list is made by using some existing variable order scheme.
- In GACCC-op we propose a new variable ordering scheme.
  - The variable, which is present in the constraint with the lowest arity.
  - Has the largest number of domain values.

- In GACCC during consistency search of a domain value of a variable, the tuples generated will contain all the variable in the conjunction set.
- In GACCC-op the consistency search for a variable x
- Will begin with tuples which contain only variables from the smallest constraint(Cs)
- Cs should contain the variable x.

GACCC & GACCC-op

- In GACCC the support list is made by using some existing variable order scheme.
- In GACCC-op we propose a new variable ordering scheme.
  - The variable, which is present in the constraint with the lowest arity.
  - Has the largest number of domain values.

- In GACCC during consistency search of a domain value of a variable, the tuples generated will contain all the variable in the conjunction set.
- In GACCC-op the consistency search for a variable x
- Will begin with tuples which contain only variables from the smallest constraint(Cs)
- Cs should contain the variable x.

Heuristic for grouping constraints into conjunctive sets

- 1. Initially there will be 'n' conjunctive sets(S), each containing a single constraint (where n is the total number of constraints in the CSP).

- 2. If there exist two conjunctive sets S1, S2 such that variables in S1 is equal to variables in S2, then remove S1 and S2 and add a new set which is conjunction of all the constraints in S1 and S2.

- 3. If there exist two conjunctive sets S1, S2 such that
  - S1, S2 share at least i variables
  - The number of variables in S1 [S2 is less than j
  - The total number of constraints in S1 and S2 is less than k
  - Remove S1 and S2 and add a new set which is conjunction of all the constraints in S1 and S2.

- 4. Repeat 2 and 3 until no more such pairs exists

Heuristic for grouping constraints into conjunctive sets

- 1. Initially there will be 'n' conjunctive sets(S), each containing a single constraint (where n is the total number of constraints in the CSP).

- 2. If there exist two conjunctive sets S1, S2 such that variables in S1 is equal to variables in S2, then remove S1 and S2 and add a new set which is conjunction of all the constraints in S1 and S2.

- 3. If there exist two conjunctive sets S1, S2 such that
  - S1, S2 share at least i variables
  - The number of variables in S1 [S2 is less than j
  - The total number of constraints in S1 and S2 is less than k
  - Remove S1 and S2 and add a new set which is conjunction of all the constraints in S1 and S2.

- 4. Repeat 2 and 3 until no more such pairs exists

Heuristic for grouping constraints into conjunctive sets

- 1. Initially there will be 'n' conjunctive sets(S), each containing a single constraint (where n is the total number of constraints in the CSP).
- 2. If there exist two conjunctive sets S1, S2 such that variables in S1 is equal to variables in S2, then remove S1 and S2 and add a new set which is conjunction of all the constraints in S1 and S2.
- 3. If there exist two conjunctive sets S1, S2 such that
  - S1, S2 share at least i variables
  - The number of variables in S1 and S2 is less than j
  - The total number of constraints in S1 and S2 is less than k
  - Remove S1 and S2 and add a new set which is conjunction of all the constraints in S1 and S2.
- 4. Repeat 2 and 3 until no more such pairs exists

Heuristic for grouping constraints into conjunctive sets

- 1. Initially there will be 'n' conjunctive sets(S), each containing a single constraint (where n is the total number of constraints in the CSP).

- 2. If there exist two conjunctive sets S1, S2 such that variables in S1 is equal to variables in S2, then remove S1 and S2 and add a new set which is conjunction of all the constraints in S1 and S2.

- 3. If there exist two conjunctive sets S1, S2 such that
  - S1, S2 share at least i variables
  - The number of variables in S1 and S2 is less than j
  - The total number of constraints in S1 and S2 is less than k
  - Remove S1 and S2 and add a new set which is conjunction of all the constraints in S1 and S2.

- 4. Repeat 2 and 3 until no more such pairs exists

Heuristic for grouping constraints into conjunctive sets

- 1. Initially there will be 'n' conjunctive sets(S), each containing a single constraint (where n is the total number of constraints in the CSP).

- 2. If there exist two conjunctive sets S1, S2 such that variables in S1 is equal to variables in S2, then remove S1 and S2 and add a new set which is conjunction of all the constraints in S1 and S2.

- 3. If there exist two conjunctive sets S1, S2 such that
  - S1, S2 share at least i variables
  - The number of variables in S1 and S2 is less than j
  - The total number of constraints in S1 and S2 is less than k
  - Remove S1 and S2 and add a new set which is conjunction of all the constraints in S1 and S2.

- 4. Repeat 2 and 3 until no more such pairs exists

Heuristic for grouping constraints into conjunctive sets

- 1. Initially there will be 'n' conjunctive sets(S), each containing a single constraint (where n is the total number of constraints in the CSP).

- 2. If there exist two conjunctive sets S1, S2 such that variables in S1 is equal to variables in S2, then remove S1 and S2 and add a new set which is conjunction of all the constraints in S1 and S2.

- 3. If there exist two conjunctive sets S1, S2 such that
  - S1, S2 share at least i variables
  - The number of variables in S1 and S2 is less than j
  - The total number of constraints in S1 and S2 is less than k
  - Remove S1 and S2 and add a new set which is conjunction of all the constraints in S1 and S2.

- 4. Repeat 2 and 3 until no more such pairs exists

Heuristic for grouping constraints into conjunctive sets

- 1. Initially there will be 'n' conjunctive sets(S), each containing a single constraint (where n is the total number of constraints in the CSP).

- 2. If there exist two conjunctive sets S1, S2 such that variables in S1 is equal to variables in S2, then remove S1 and S2 and add a new set which is conjunction of all the constraints in S1 and S2.

- 3. If there exist two conjunctive sets S1, S2 such that
  - S1, S2 share at least i variables
  - The number of variables in S1 [S2 is less than j
  - The total number of constraints in S1 and S2 is less than k
  - Remove S1 and S2 and add a new set which is conjunction of all the constraints in S1 and S2.

- 4. Repeat 2 and 3 until no more such pairs exists

Correctness of Algorithm

In order to prove the correctness of the algorithm we proved the following:

- Algorithm will terminate.
- The algorithm does not remove any consistent value from the domain of variables.
- The algorithm will not miss any valid tuple during the generation of next tuple
- When the algorithm terminates, then the domain of variables contain only arc consistent values (or some domain is empty).

- Time Complexity=$O(en^2d^n)$
- Space Complexity=$O(en^2d)$

e= no: of constraints n=no: of variables d=highest no: of domain

Correctness of Algorithm

In order to prove the correctness of the algorithm we proved the following:

- Algorithm will terminate.
- The algorithm does not remove any consistent value from the domain of variables.
- The algorithm will not miss any valid tuple during the generation of next tuple
- When the algorithm terminates, then the domain of variables contain only arc consistent values (or some domain is empty).

- Time Complexity=O(**en^2d^n**)
- Space Complexity=O(**en^2d**)

  e= no: of constraints n=no: of variables d=highest no: of domain

Correctness of Algorithm

In order to prove the correctness of the algorithm we proved the following:

- Algorithm will terminate.
- The algorithm does not remove any consistent value from the domain of variables.
- The algorithm will not miss any valid tuple during the generation of next tuple
- When the algorithm terminates, then the domain of variables contain only arc consistent values (or some domain is empty).

- Time Complexity=O(**en²dⁿ**)
- Space Complexity=O(**en²d**)

e= no: of constraints n=no: of variables d=highest no: of domain

Correctness of Algorithm

In order to prove the correctness of the algorithm we proved the following:

- Algorithm will terminate.
- The algorithm does not remove any consistent value from the domain of variables.
- The algorithm will not miss any valid tuple during the generation of next tuple
- When the algorithm terminates, then the domain of variables contain only arc consistent values (or some domain is empty).

- Time Complexity=O(en^2d^n)
- Space Complexity=O(en^2d)

  e= no: of constraints n=no: of variables d=highest no: of domain

Correctness of Algorithm

In order to prove the correctness of the algorithm we proved the following:

- Algorithm will terminate.
- The algorithm does not remove any consistent value from the domain of variables.
- The algorithm will not miss any valid tuple during the generation of next tuple
- When the algorithm terminates, then the domain of variables contain only arc consistent values (or some domain is empty).

- Time Complexity=O($\mathbf{en^2d^n}$)
- Space Complexity=O($\mathbf{en^2d}$)

  e= no: of constraints n=no: of variables d=highest no: of domain

Experimental Results

- Comparison with existing GAC algorithm
- 3-Sat Problem with binary domain(0,1)

No: of Variables	No: of Constraints	No of tuples for GACCC	No of tuples for GACCC-2	% improvement in time
10	14	98	76	12.34
12	14	96	70	10.66
14	14	103	82	11.46
18	30	168	120	19.86
20	30	170	131	17.96
20	40	256	216	17.43

Experimental Results

- Proposed algorithm used with VCS(a CDTG tool)

			Improvement After Domain Reduction	
Bench mark Problems	No: of variables	No: of Domain values	Time (%)	Memory (%)
Langford Series	6	3	10.0	23.5
	8	4	21.4	27.7
	14	7	25.0	40.8
Golomb Ruler	3	4	8.3	23.2
	4	7	7.1	28.2
	5	12	9.5	39.1
	6	18	13.8	73.1
Magic Sequence	4	4	30	50.0
	5	5	40	71.6
	7	7	55	73.3
	8	8	62.5	81.5

Conclusion & Future Work

CONCLUSION

- Presented a new consistency check algorithm.
- The algorithm reduce the memory used and time required to generate the test cases.

FUTURE WORK

- Use consistency algorithm for domain clustering to have uniformity in randomization.
- Attain 100% coverage in few iterations.

Conclusion & Future Work

CONCLUSION

- Presented a new consistency check algorithm.
- The algorithm reduce the memory used and time required to generate the test cases.

FUTURE WORK

- Use consistency algorithm for domain clustering to have uniformity in randomization.
- Attain 100% coverage in few iterations.

Conclusion & Future Work

CONCLUSION

- Presented a new consistency check algorithm.
- The algorithm reduce the memory used and time required to generate the test cases.

FUTURE WORK

- Use consistency algorithm for domain clustering to have uniformity in randomization.
- Attain 100% coverage in few iterations.

Conclusion & Future Work

CONCLUSION

- Presented a new consistency check algorithm.
- The algorithm reduce the memory used and time required to generate the test cases.

FUTURE WORK

- Use consistency algorithm for domain clustering to have uniformity in randomization.
- Attain 100% coverage in few iterations.

Questions & Answers