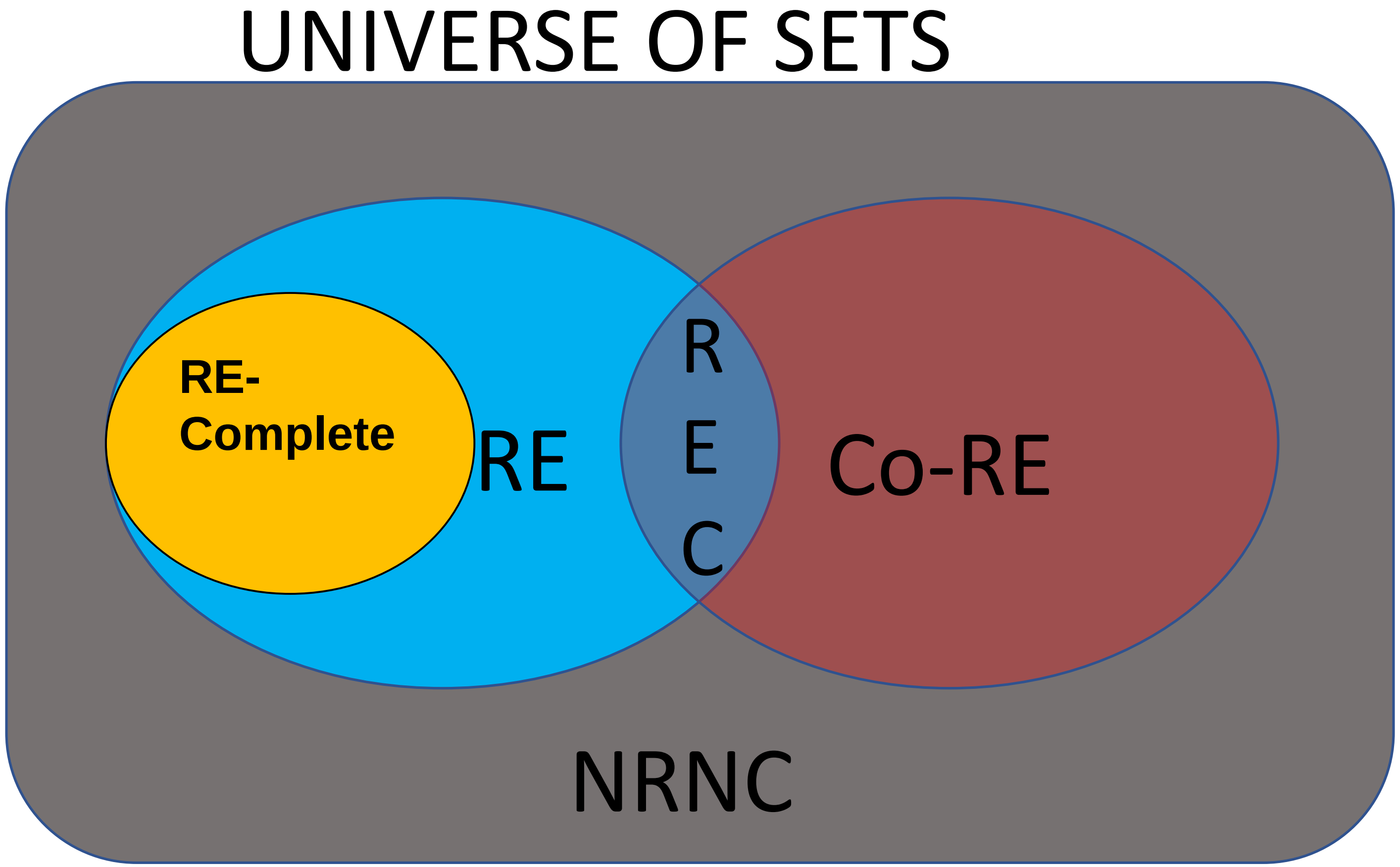


NR (non-recursive)

= (NRNC U Co-RE) - REC

Some Quantification Examples

- $\langle f,x \rangle \in Halt \Leftrightarrow \exists t\, [\, STP(f,x,t)\,]$ RE
- $f \in Total \Leftrightarrow \forall x \exists t\, [\, STP(f,x,t)\,]$ NRNC
- $f \in NotTotal \Leftrightarrow \exists x \forall t\, [\, \sim STP(f,x,t)\,]$ NRNC
- $f \in RangeAll \Leftrightarrow \forall x \exists \langle y,t \rangle\, [\, STP(f,y,t)\, \&\, VALUE(f,y,t)=x\,]$ NRNC
- $f \in RangeNotAll \Leftrightarrow \exists x \forall \langle y,t \rangle\, [STP(f,y,t) \Rightarrow VALUE(f,y,t) \neq x\,]$ NRNC
- $f \in HasZero \Leftrightarrow \exists \langle x,t \rangle\, [\, STP(f,x,t)\, \&\, VALUE(f,x,t)=0\,]$ RE
- $f \in IsZero \Leftrightarrow \forall x \exists t\, [\, STP(f,x,t)\, \&\, VALUE(f,x,t)=0\,]$ NRNC
- $f \in Empty \Leftrightarrow \forall \langle x,t \rangle\, [\, \sim STP(f,x,t)\,]$ Co-RE
- $f \in NotEmpty \Leftrightarrow \exists\, \langle x,t \rangle\, [\, STP(f,x,t)\,]$ RE

More Quantification Examples

- $f \in \text{Identity} \Leftrightarrow \forall x \exists t\, [\, STP(f,x,t)\ \&\ VALUE(f,x,t)=x \,]$ NRNC
- $f \in \text{NotIdentity} \Leftrightarrow \exists x \forall t\, [\, \sim STP(f,x,t) \mid VALUE(f,x,t) \neq x \,]$ or
  $\exists x \forall t\, [\, STP(f,x,t) \Rightarrow VALUE(f,x,t) \neq x \,]$ NRNC
- $f \in \text{Constant} = \forall \langle x,y \rangle \exists t\, [STP(f,x,t)\ \&\ STP(f,y,t)\ \&$
  $VALUE(f,x,t)=VALUE(f,y,t)]$ NRNC
- $f \in \text{Infinite} \Leftrightarrow \forall x \exists \langle y,t \rangle\, [\, y \geq x\ \&\ STP(f,y,t) \,]$ NRNC
- $f \in \text{Finite} \Leftrightarrow \exists x \forall \langle y,t \rangle\, [\, y < x \mid \sim STP(f,y,t) \,]$ or
  $\exists x \forall \langle y,t \rangle\, [\, STP(f,y,t) \Rightarrow y < x \,]$ or $[\, y \geq x \Rightarrow \sim STP(f,y,t) \,]$ NRNC
- $f \in \text{RangeInfinite} \Leftrightarrow \forall x \exists \langle y,t \rangle\, [\, STP(f,y,t)\ \&\ VALUE(f,y,t) \geq x \,]$ NRNC
- $f \in \text{RangeFinite} \Leftrightarrow \exists x \forall \langle y,t \rangle\, [\, STP(f,y,t) \Rightarrow VALUE(f,y,t) < x \,]$ NRNC
- $f \in \text{Stutter} \Leftrightarrow \exists \langle x,y,t \rangle\, [\, x \neq y\ \&\ STP(f,x,t)\ \&\ STP(f,y,t)\ \&$
  $VALUE(f,x,t) = VALUE(f,y,t) \,]$ RE

Even More Quantification Examples

$\langle f,x\rangle \in \text{Fast20} \Leftrightarrow [\ \text{STP}(f,x,20)\ ]$	REC
$f \in \text{FastOne20} \Leftrightarrow \exists x\ [\ \text{STP}(f,x,20)\ ]$	RE
$f \in \text{FastAll20} \Leftrightarrow \forall x\ [\ \text{STP}(f,x,20)\ ]$	Co-RE
$\langle f,x,K,C\rangle \in \text{LinearKC} \Leftrightarrow [\ \text{STP}(f,x,K*x+C)\ ]$	REC
$\langle f,K,C\rangle \in \text{LinearKCOne} \Leftrightarrow \exists x\ [\ \text{STP}(f,x,K*x+C)\ ]$	RE
$\langle f,K,C\rangle \in \text{LinearKCAll} \Leftrightarrow \forall x\ [\ \text{STP}(f,x,K*x+C)\ ]$	Co-RE

- None of the above can be shown undecidable using Rice’s Theorem
- In fact, reduction from known undecidables is also a problem for all but the first one which happens to be decidable.

Some Reductions and Rice Example

- **NotEmpty ≤ Halt**
  Let f be an arbitrary index
  Define $\forall y\ g_f(y) = \exists$<x,t> STP(f,x,t)
  f $\in$ ENotmpty $\Leftrightarrow$ <g_f,0> $\in$ Halt
- **Halt ≤ NotEmpty**
  Let f,x be an arbitrary index and input value
  Define $\forall y\ g_{f,x}(y) = f(x)$
  <f,x> $\in$ Halt$\Leftrightarrow$ $g_{f,x}$ $\in$ Empty
- **Note: NotEmpty is RE-Complete**
- **Rice: NotEmpty is non-trivial Zero$\in$NotEmpty; $\uparrow\notin$NotEmpty**
  Let f,g be arbitrary indices such that Dom(f)=Dom(g)
  f $\in$NotEmpty $\Leftrightarrow$ Dom(f) $\neq \emptyset$ By Definition
  $\Leftrightarrow$ Dom(g) $\neq \emptyset$ Dom(g)=Dom(f)
  $\Leftrightarrow$ g $\in$NotEmpty
  Thus, Rice's Theorem states that NotEmpty is undecidable.

More Reductions and Rice Example

- **Identity ≤ Total**
  Let f be an arbitrary index
  Define $g_f(x) = \mu y\ [\ f(x) = x\ ]$
  $f \in$ Identity $\Leftrightarrow g_f \in$ Total
- **Total ≤ Identity**
  Let f be an arbitrary index
  Define $g_f(x) = f(x)-f(x) + x$
  $f \in$ Total $\Leftrightarrow g_{f,x} \in$ Identity
- **Rice: Identity is non-trivial $I(x)=x \in$ Identity; Zero $\notin$ Identity**
  Let f,g be arbitrary indices such that $\forall x\ f(x) = g(x)$
  $f \in$ Identity $\Leftrightarrow \forall x\ f(x)=x$ By Definition
  $\Leftrightarrow \forall x\ g(x)=x$ $\forall x\ g(x) = f(x)$
  $\Leftrightarrow g \in$ Identity
  Thus, Rice's Theorem states that Identity is undecidable

Even More Reductions and Rice Example

- **Stutter ≤ Halt**
  Let f be an arbitrary index
  Define $\forall y\ g_f(y) = \exists\langle x,y,t\rangle\ [\ x\neq y\ \&\ STP(f,x,t)\ \&\ STP(f,y,t)\ \&$
  $VALUE(f,x,t) = VALUE(f,y,t)\ ]$
  $f \in Stutter \Leftrightarrow \langle g_f,0\rangle \in Halt$
- **Halt ≤ Stutter**
  Let f,x be an arbitrary index and input value
  Define $\forall y\ g_{f,x}(y) = f(x)$
  $\langle f,x\rangle \in Halt \Leftrightarrow g_{f,x} \in Stutter$
- **Note: Stutter is RE-Complete**
- **Rice: Stutter is non-trivial Zero∈Stutter; $I(x)=x \notin Stutter$**
  Let f,g be arbitrary indices such that $\forall x\ f(x) = g(x)$
  $f \in Stutter \Leftrightarrow \exists\langle x,y\rangle\ [\ x\neq y\ \&\ f(x)=f(y)\ ]$ **By Definition**
  $\Leftrightarrow \exists\langle x,y\rangle\ [\ x\neq y\ \&\ g(x)=g(y)\ ]$ **$\forall x\ g(x) = f(x)$**
  $\Leftrightarrow g \in Stutter$
  Thus, Rice's Theorem states that Identity is undecidable

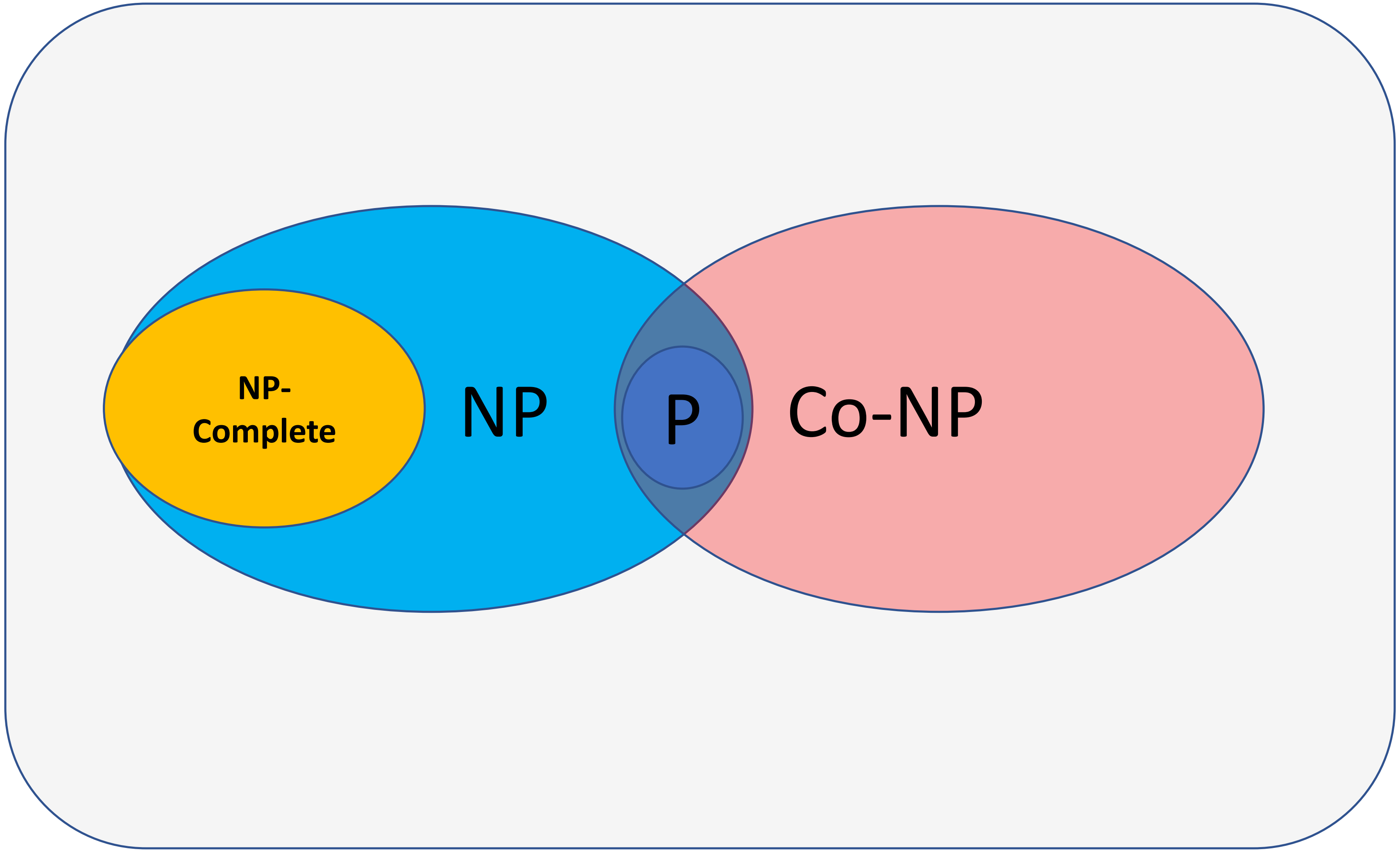
Yet More Reductions and Rice Example

- **Constant ≤ Total**
  Let f be an arbitrary index
  Define $g_f(0) = f(0)$
  $g_f(y+1) = \mu y\ [\ f(y+1) = f(y)\ ]$
  $f \in$ Constant $\Leftrightarrow g_f \in$ Total
- **Total ≤ Identity**
  Let f be an arbitrary index
  Define $g_f(x) = f(x)-f(x)$
  $f \in$ Total $\Leftrightarrow g_f \in$ Constant
- **Rice: Constant is non-trivial Zero∈Constant; $I(x)=x \notin$ Constant**
  Let f,g be arbitrary indices such that $\forall x\ f(x) = g(x)$
  $f \in$Constant $\Leftrightarrow$ $\exists C \forall x\ f(x)=C$ By Definition
  $\Leftrightarrow$ $\exists C \forall x\ g(x)=C$ $\forall x\ g(x) = f(x)$
  $\Leftrightarrow g \in$Constant
  Thus, Rice's Theorem states that Identity is undecidable

Last Reductions and Rice Example

- **RangeAll ≤ Total**
  Let f be an arbitrary index
  Define $g_f(x) = \exists y\, [\, f(y) = x\,]$
  $f \in$ RangeAll $\Leftrightarrow g_f \in$ Total
- **Total ≤ RangeAll**
  Let f be an arbitrary index
  Define $g_f(x) = f(x)-f(x) + x$
  $f \in$ Total $\Leftrightarrow g_f \in$ RangeAll
- **Rice: RangeAll is non-trivial $I(x)=x \in$ RangeAll; Zero $\notin$ RangeAll**
  Let f,g be arbitrary indices such that Range(f) = Range(g)
  $f \in$ RangeAll $\Leftrightarrow$ Range(f) = $\aleph$ By Definition
  $\Leftrightarrow$ Range(f) = $\aleph$ Range(g) = Range(f)
  $\Leftrightarrow g \in$ RangeAll
  Thus, Rice's Theorem states that Identity is undecidable

UNIVERSE OF SETS



Complexity Sample#1

#	Concept	Description	Concept #
1	Problem A is in NP	The classic NP-Complete problem	10
2	Problem A is in co-NP	A is the problem TOTAL (set of Algorithms)	4
3	Problem A is in P	A is decidable in deterministic polynomial time	3
4	Problem A is non-RE/non-Co-RE	If B is in NP then $B \leq_P A$	9
5	Problem A is NP-Complete	A is in RE and, if B is in RE, then $B \leq_m A$	8
6	Problem A is RE	A is verifiable in deterministic polynomial time	1
7	Problem A is Co-RE	A is in NP and if B is in NP then $B \leq_P A$	5
8	Problem A is RE-Complete	A is semi-decidable	6
9	Problem A is NP-Hard	A is the complement of B and B is RE	7
10	Satisfiability	A’s complement is in NP	2

Sample#2: 3SAT to SubsetSum

(~a + b + ~c) (~a + ~b + c)

	a	b	c	~a + b + ~c	~a + ~b + c
a	1	0	0	0	0
~a	1	0	0	1	1
b	0	1	0	1	0
~b	0	1	0	0	1
c	0	0	1	0	1
~c	0	0	1	1	0
C1	0	0	0	1	0
C1’	0	0	0	1	0
C2	0	0	0	0	1
C2’	0	0	0	0	1
	1	1	1	3	3

Sample#3: Scheduling

List Schedule (T1,4), (T2,5), (T3,2), (T4,7), (T5,1), (T6,4), (T7,8)

T1	**T1**	**T1**	**T1**	**T3**	**T3**	**T5**	**T6**	**T6**	**T6**	**T6**	**T7**	**T7**	**T7**	**T7**	**T7**	**T7**	**T7**	**T7**
T2	T2	T2	T2	T2	T4	T4	T4	T4	T4	T4	T4							

Sorted List Schedule (T7,8), (T4,7), (T2,5), (T1,4), (T6,4), (T3,2), (T5,1)

T7	**T7**	**T7**	**T7**	**T7**	**T7**	**T7**	**T7**	**T1**	**T1**	**T1**	**T1**	**T6**	**T6**	**T6**	**T6**			
T4	T4	T4	T4	T4	T4	T4	T2	T2	T2	T2	T2	T3	T3	T5				

Independent set (IS) is NP-Complete

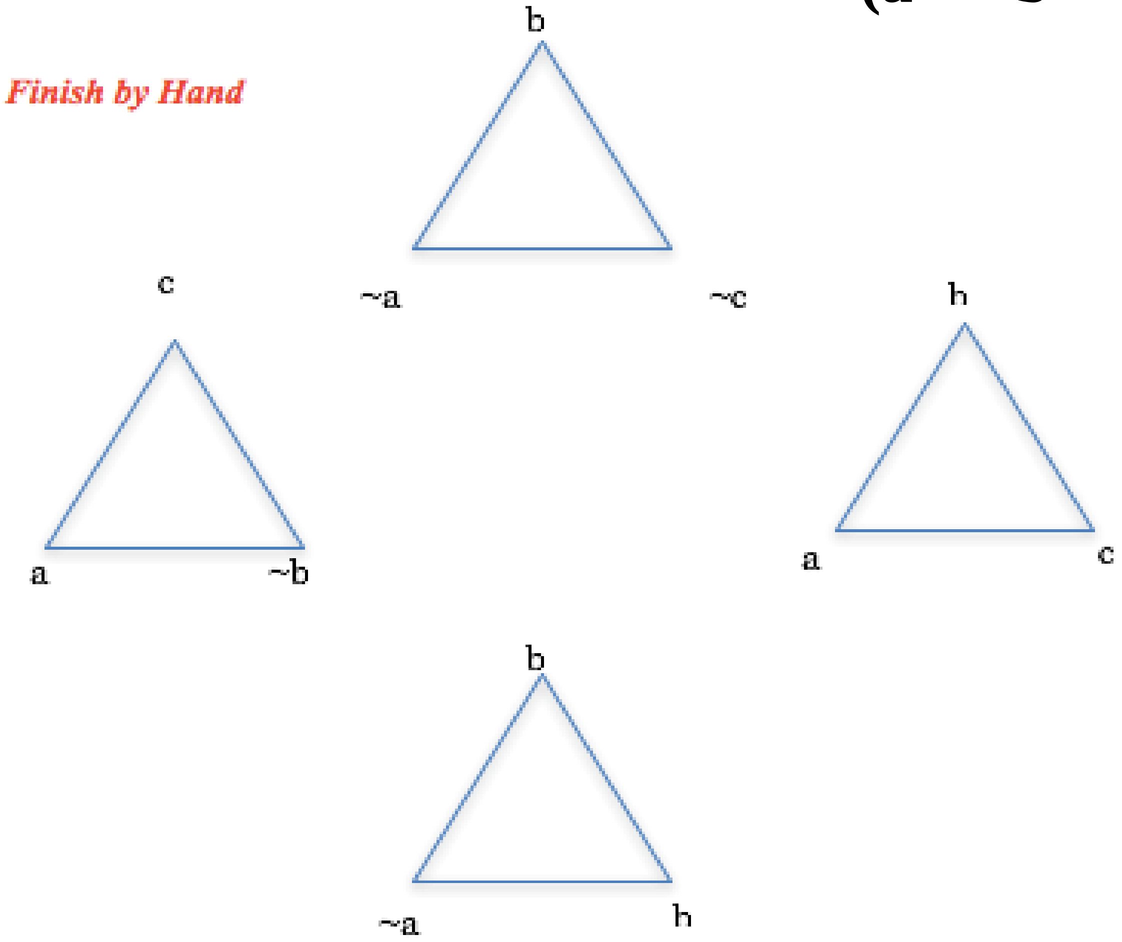
- We represent each clause in an instance of 3SAT with a triangle, one node per literal. The key is that all nodes are connected in a triangle of nodes, so the best you can do is to choose one node per clause to participate in an independent set. By adding an edge between every instance of variable v and every instance of variable ~v, we guarantee that we cannot choose nodes labeled v and ~v as part of an independent set. Here, assume we have V Boolean variables
- When the required independent set must be C, where C is the number of clauses, we must choose one node per clause and we must do this in a way so that no nodes labeled with a variable and its complement are chosen. That can only be done if there is an assignment to variables (true or false) that satisfy the original instance of 3SAT. Thus IS is NP-Hard. But, we can check a proposed independent set in time proportional to the size of the graph (which is actually linear in the size of the 3SAT problem). Thus IS is in P. In conclusion, IS is NP-Complete.

Sample#4: Independent Set

k=4

Finish by Hand

(a + ~b + c) (~a + b + ~c) (a + b + c) (~a + b + b)

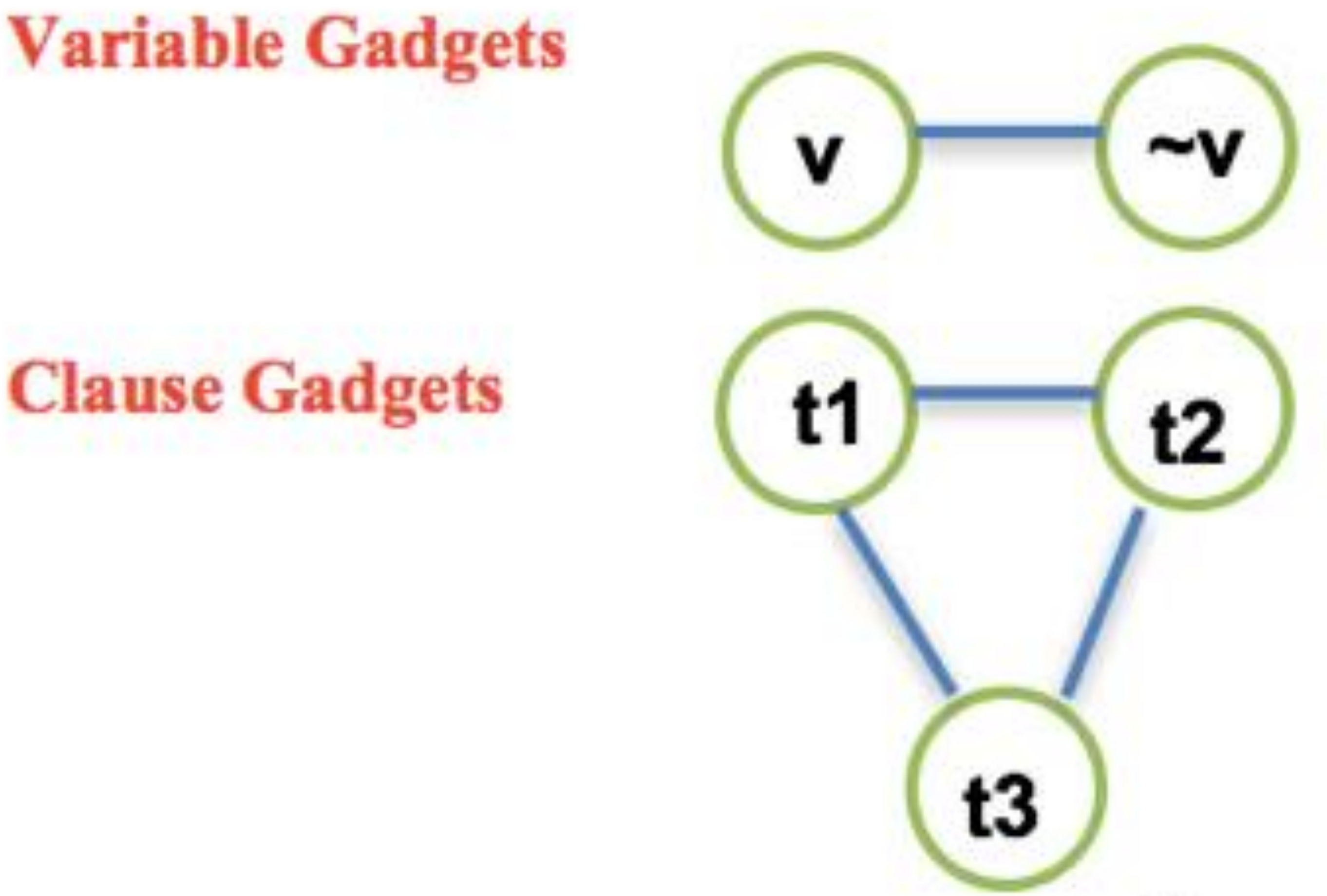


Place an edge between every node labeled V and every node labeled ~V, where V can be a, b or c.

Vertex Cover (VC) is NP-Complete

- We represent each clause (assume there are C of them) in an instance of 3SAT with a triangle, one node per literal. One key is that two nodes in each clause triangle must be chosen to cover the three internal edges. We represent each assignment to a variable v (assume there are V variables) by a pair of connected nodes labeled v and ~v. The second key is that we must choose precisely one of v or ~v for each variable to cover the edge that connects its pair. Thus, the minimum cover set contains 2C+V nodes.
- We add an edge from each v and to all literals v in clauses, and each ~v to all literals ~v in clauses. To cover all the edges added here for the variable nodes, we must choose nodes in each clause that cover edges from variable nodes that are not chosen in the variable pair. If all clauses have at least one of these incoming edges already covered (we chose an assignment to the variable that matches a literal in this clause), then we will be able to cover all internal edges in each clause and all edges entering the clause from a variable pair, by just choosing two nodes in the clause.
- Choosing 2C+V nodes that cover all edges can only be done if there is an assignment to variables (true or false) that satisfy the original instance of 3SAT. Thus VC is NP-Hard. But, we can check a proposed cover set of vertices in time proportional to the size of the graph (which is actually linear in the size of the 3SAT problem). Thus VC is in P. In conclusion, VC is NP-Complete.

Sample # 5: VC Gadgets

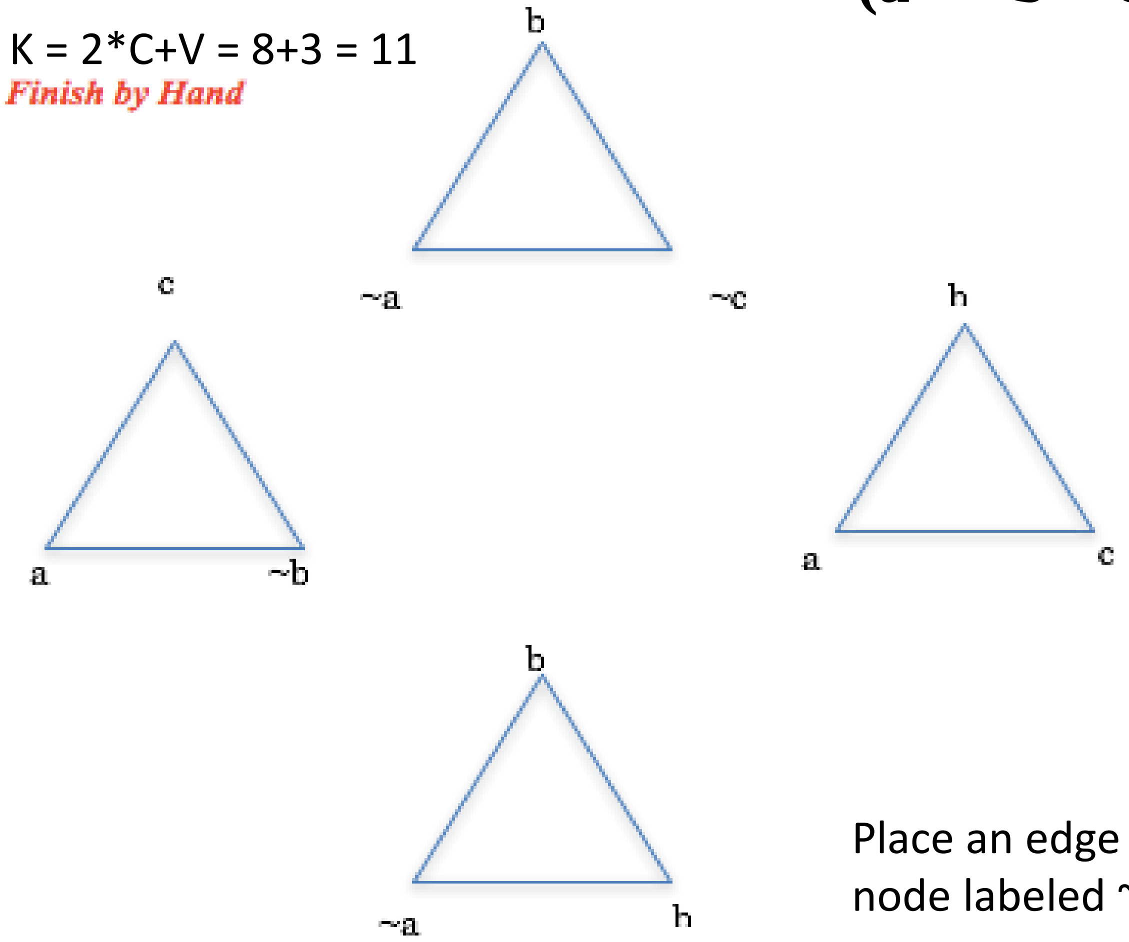


Sample#6: Vertex Cover

K = 2*C+V = 8+3 = 11

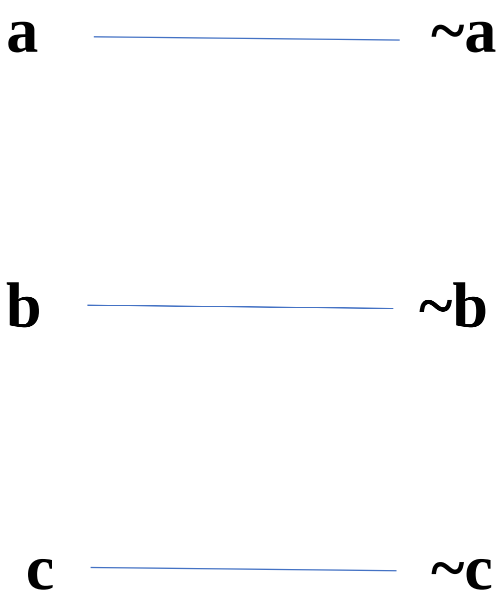
Finish by Hand

Clause Nodes/Edges



(a + ~b + c) (~a + b + ~c) (a + b + c) (~a + b + b)

Variable Nodes/Edges



Place an edge between every variable node labeled V and every clause node labeled ~V, where V can be a, b or c.