Automata for Associative-Commutative Operators

Kumar Neeraj Verma

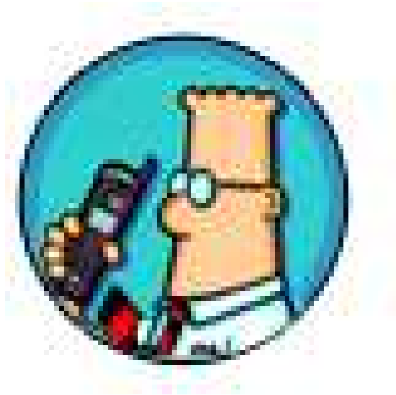
TU München

Motivations

- Cryptographic protocols
- XML documents
- Petri nets
- Linear logic

Public-key Needham-Schroeder protocol

Alice

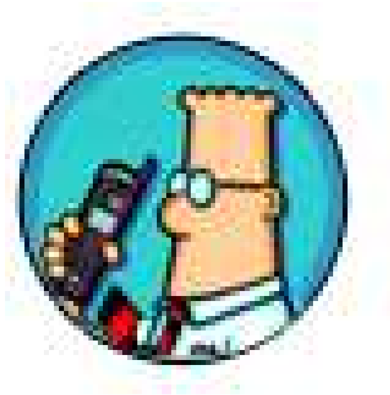


Bob

Public-key Needham-Schroeder protocol

Alice

$$\{Alice, Random_1\}_{pub(Bob)}$$



Bob

Public-key Needham-Schroeder protocol

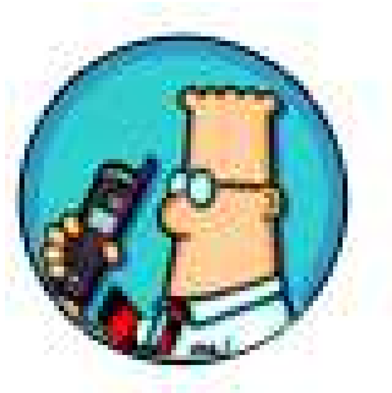
Alice

$$\{Alice, Random_1\}_{pub(Bob)}$$

$$\longrightarrow$$

$$\{Random_1, Random_2\}_{pub(Alice)}$$

$$\longleftarrow$$



Bob

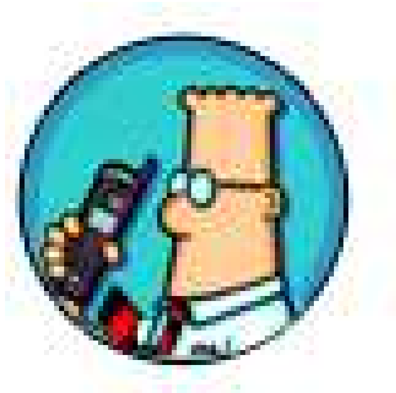
Public-key Needham-Schroeder protocol

Alice

$$\{Alice, Random_1\}_{pub(Bob)}$$

$$\{Random_1, Random_2\}_{pub(Alice)}$$

$$\{Random_2\}_{pub(Bob)}$$



Bob

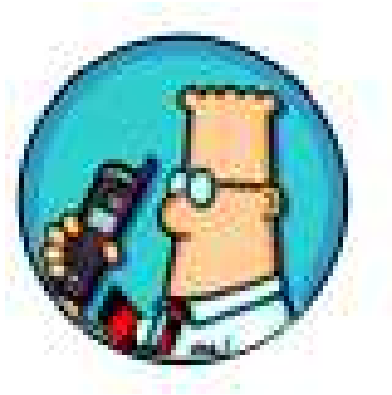
Public-key Needham-Schroeder protocol

Alice

$$\{Alice, Random_1\}_{pub(Bob)} \longrightarrow$$

$$\longleftarrow \{Random_1, Random_2\}_{pub(Alice)}$$

$$\{Random_2\}_{pub(Bob)} \longrightarrow$$



Bob

Attack against the protocol found after 17 years!

Cryptographic protocols and tree automata

Dolev-Yao Model [Dolev & Yao, 1983]:

message	term
nonces, identities, ...	constants
$\{m\}_k$	$\texttt{encrypt}(m, k)$
$< m_1, m_2 >$	$\texttt{pair}(m_1, m_2)$
...	...

$\Rightarrow$ Use tree automata to model intruder's knowledge.

Cryptographic protocols and tree automata

Dolev-Yao Model [Dolev & Yao, 1983]:

message	term
nonces, identities, ...	constants
$\{m\}_k$	$\texttt{encrypt}(m, k)$
$< m_1, m_2 >$	$\texttt{pair}(m_1, m_2)$
...	...

$\Rightarrow$ Use tree automata to model intruder's knowledge.

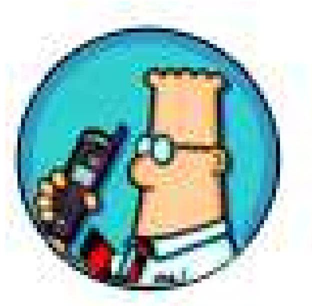
Non-ideal cryptographic primitives, e.g. modular exponentiation:

$$\alpha^{x.y} = \alpha^{y.x} \qquad (\alpha^x)^y = \alpha^{x.y} \quad \ldots$$

$\Rightarrow$ Use equational tree automata

Group key agreement protocols

The IKA.1 protocol [Steiner et al, 2000] for 3 participants:



B

C

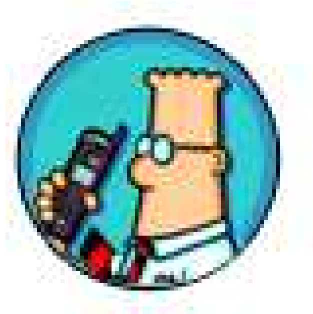
A

Group key agreement protocols

The IKA.1 protocol [Steiner et al, 2000] for 3 participants:

A

$$\xrightarrow{\alpha, \alpha^{Na}}$$

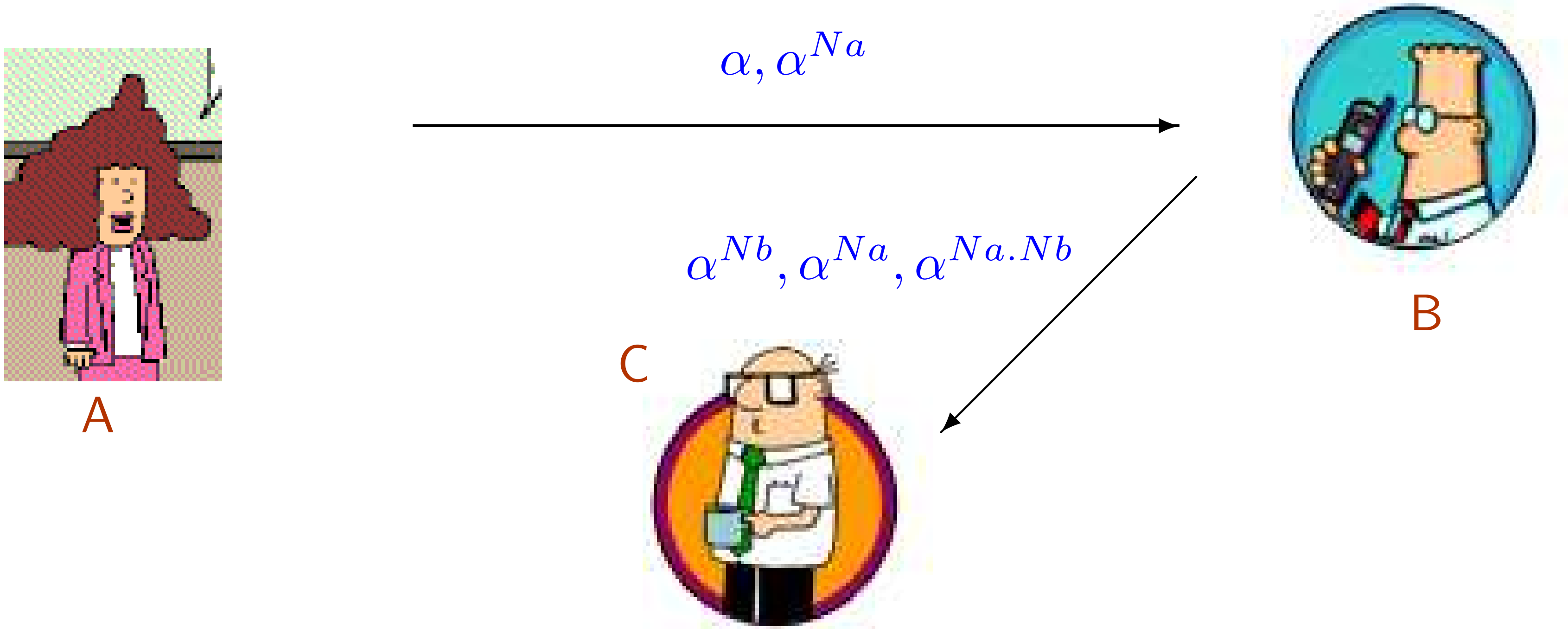


B

C

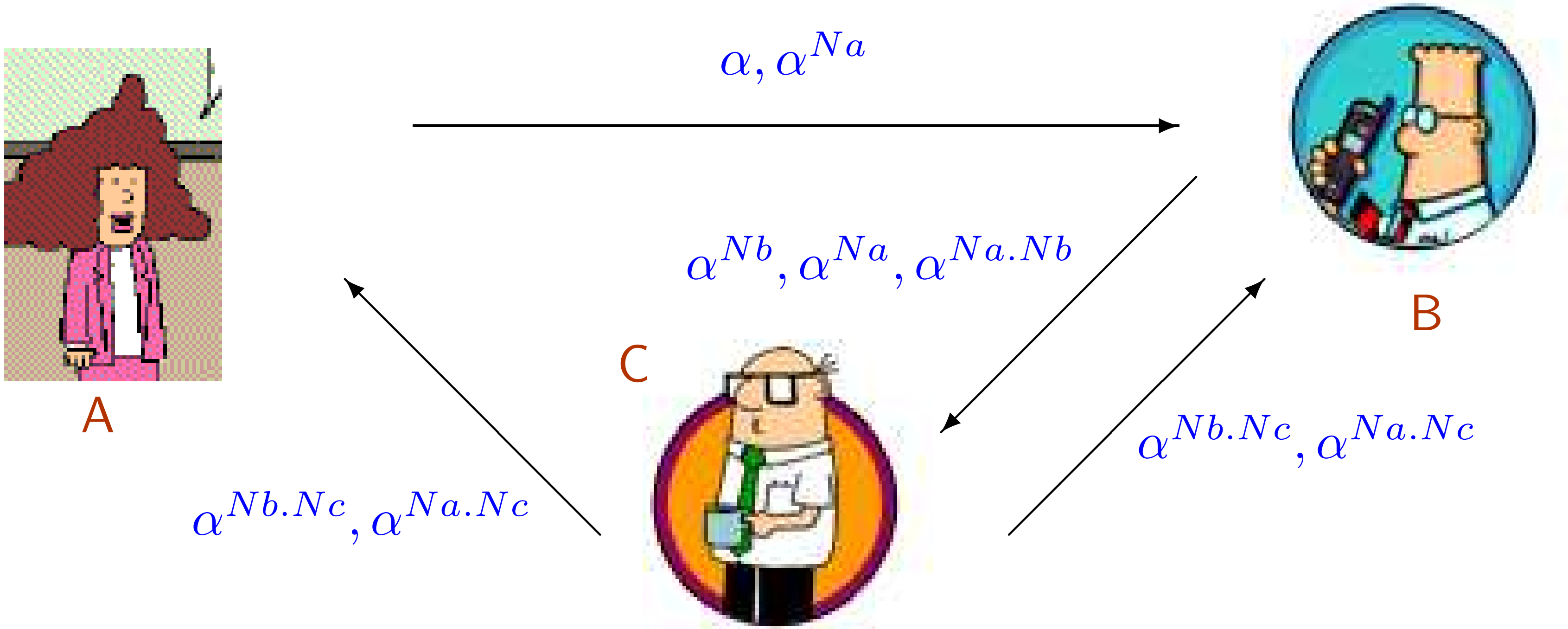
Group key agreement protocols

The IKA.1 protocol [Steiner et al, 2000] for 3 participants:



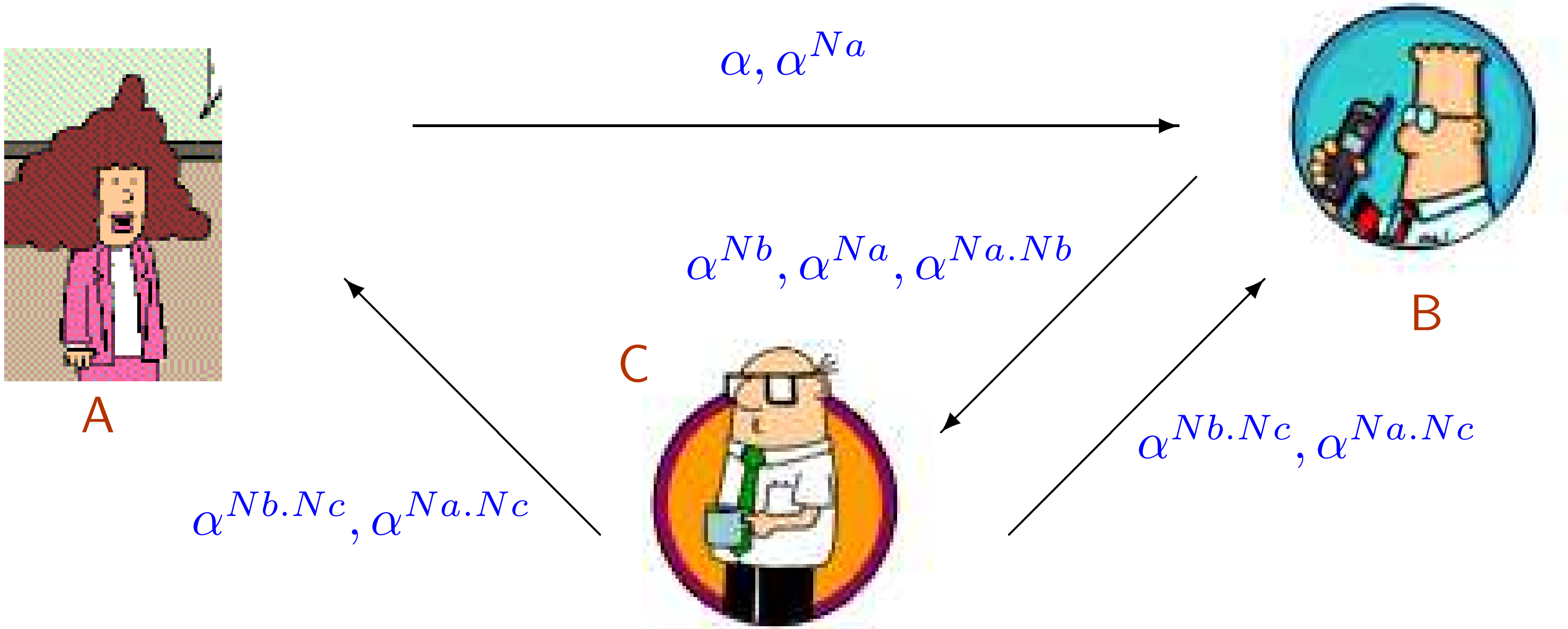
Group key agreement protocols

The IKA.1 protocol [Steiner et al, 2000] for 3 participants:



Group key agreement protocols

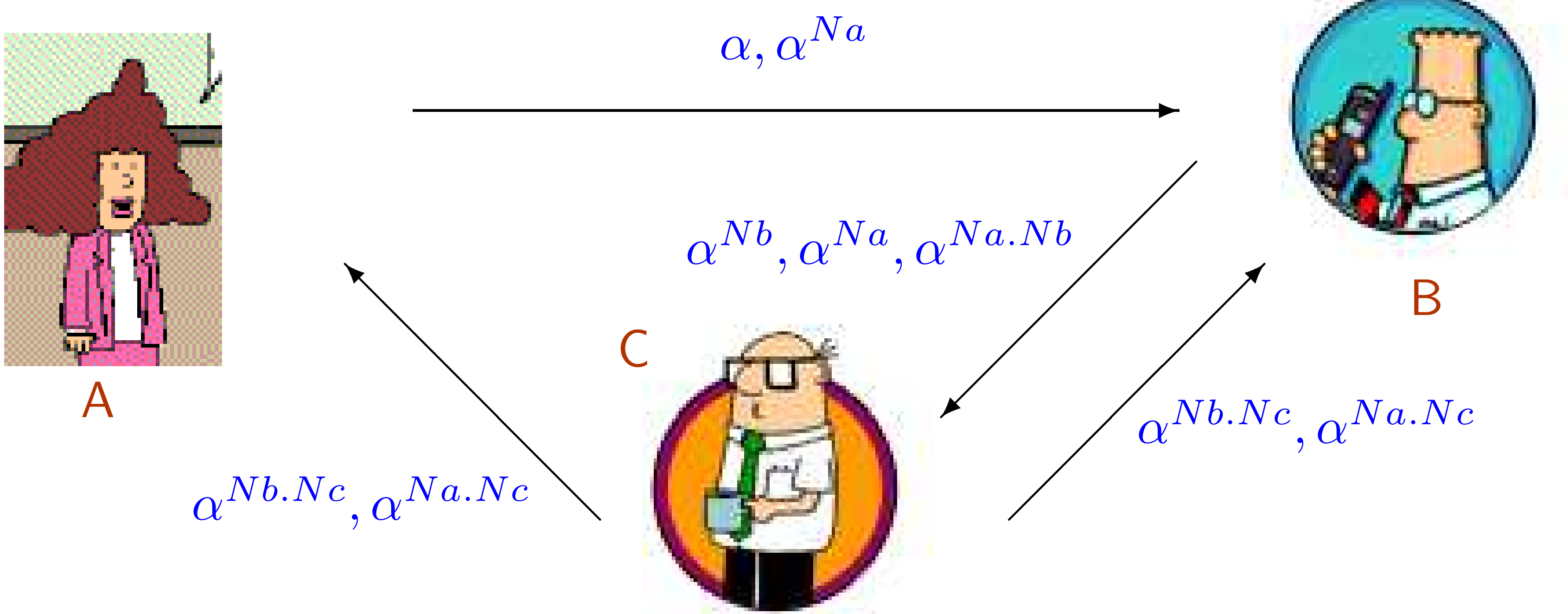
The IKA.1 protocol [Steiner et al, 2000] for 3 participants:



Group key $\alpha^{Na.Nb.Nc}$ is then computed by each participant

Group key agreement protocols

The IKA.1 protocol [Steiner et al, 2000] for 3 participants:



Group key $\alpha^{Na.Nb.Nc}$ is then computed by each participant

Code messages $\alpha^{x_1 \ldots x_n}$ by terms $e(x_1 + \ldots + x_n)$

$\Rightarrow +$ is ACU

Use automata modulo ACU

The ACU theory

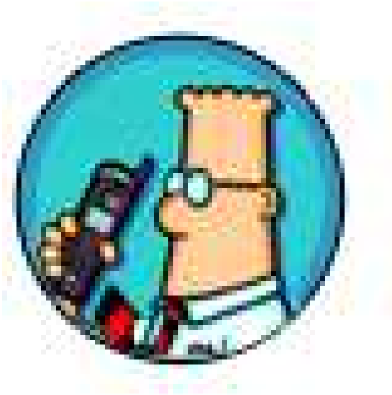
$x+(y+z)=(x+y)+z$	Associativity
$x+y=y+x$	Commutativity
$x+0=x$	Unit

A protocol using XOR

An example protocol using XOR [Cortier03]:

$+$ is XOR

Alice



Bob

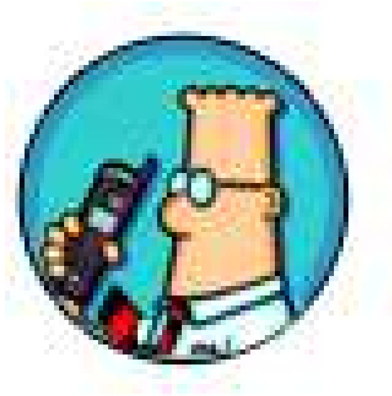
A protocol using XOR

An example protocol using XOR [Cortier03]:

$+$ is XOR

Alice

$$\xrightarrow{N_a + K_{ab}}$$



Bob

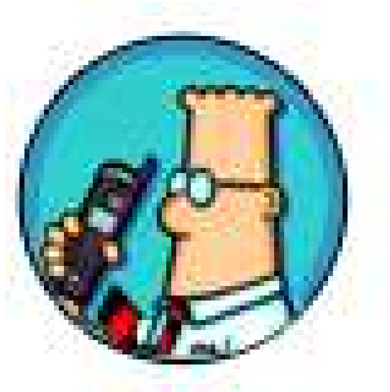
A protocol using XOR

An example protocol using XOR [Cortier03]:

$+$ is XOR

$$\xrightarrow{N_a + K_{ab}}$$

$$\xleftarrow{N_b + N_a}$$



Alice

Bob

A protocol using XOR

An example protocol using XOR [Cortier03]:

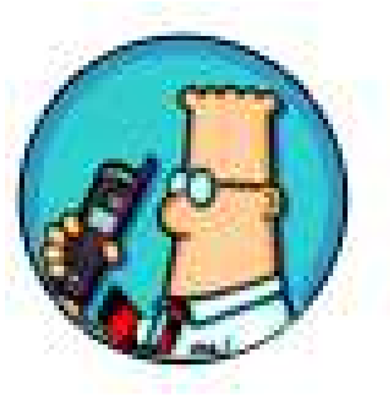
$+$ is XOR

Alice

$$\text{Alice} \xrightarrow{N_a + K_{ab}} \text{Bob}$$

$$\text{Alice} \xleftarrow{N_b + N_a} \text{Bob}$$

$$\text{Alice} \xrightarrow{S_{ab} + N_b} \text{Bob}$$



Bob

A protocol using XOR

An example protocol using XOR [Cortier03]:

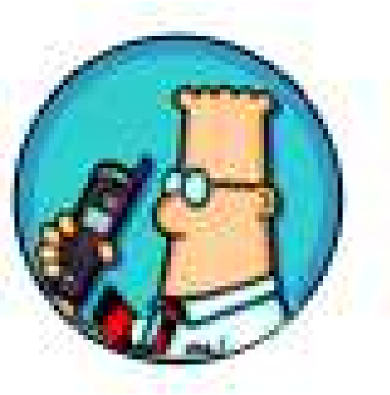
$+$ is XOR

Alice

$N_a + K_{ab}$

$N_b + N_a$

$S_{ab} + N_b$



Bob

Requires tree automata modulo XOR

The XOR theory

The ACU theory, together with the equation

$$x+x=0 \quad \text{Nilpotence}$$

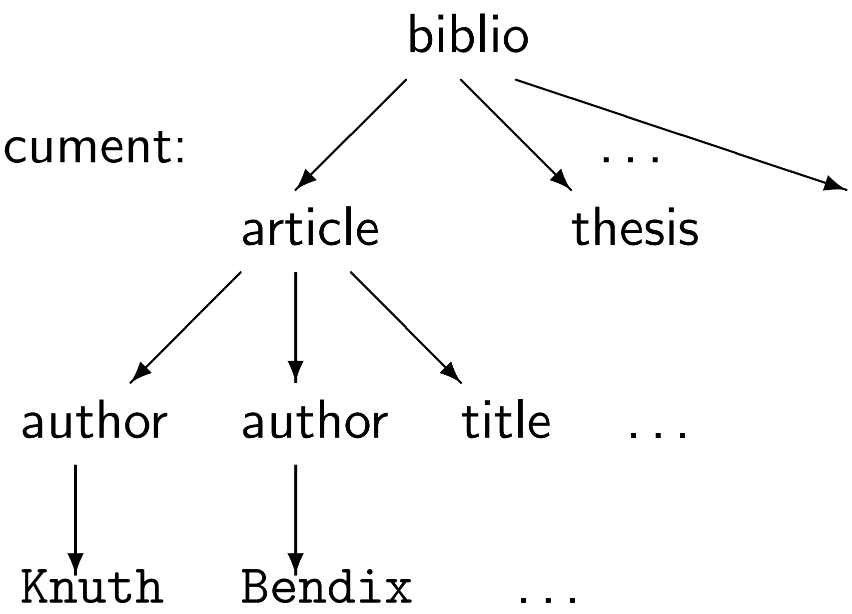
Another example: the Abelian Groups theory

The ACU theory, together with the equation

$$x+(-x)=0 \quad \text{Cancellation}$$

XML Schemas

An example XML document:



This is a term with variable arity symbols

$\Rightarrow$ Represent using associative and associative-commutative symbol

The above document is the term:

biblio (article (author (`Knuth`) + author (`Bendix`) + ...) + thesis (...) + ...)

XML Schemas $\equiv$ a class of documents $\equiv$ tree automata

Example: a bibliographic database is a list of articles, an article has a list of authors, a title, . . .

Represent using tree automata [DalZilioLugiez03, Seidl03]

An example query: search for articles written by Knuth in 1978 with odd number of coauthors

Translate to tree automata

Tree automata as Horn clauses

Automata transitions	
$\delta(O) = q_{even}$	
$\delta(S, q_{even}) = q_{odd}$	
$\delta(S, q_{odd}) = q_{even}$	
$\delta(+, q_{even}, q_{even}) = q_{even}$	
$\delta(+, q_{even}, q_{odd}) = q_{odd}$	

...

Tree automata as Horn clauses

Automata transitions		Horn clauses	
$\delta(O)$	$= q_{even}$	$q_{even}(O)$	
$\delta(S, q_{even})$	$= q_{odd}$	$q_{odd}(S(x))$	$\Leftarrow q_{even}(x)$
$\delta(S, q_{odd})$	$= q_{even}$	$q_{even}(S(x))$	$\Leftarrow q_{odd}(x)$
$\delta(+, q_{even}, q_{even})$	$= q_{even}$	$q_{even}(+(x, y))$	$\Leftarrow q_{even}(x), q_{even}(y)$
$\delta(+, q_{even}, q_{odd})$	$= q_{odd}$	$q_{odd}(+(x, y))$	$\Leftarrow q_{even}(x), q_{odd}(y)$

...

Tree automata as Horn clauses

Automata transitions		Horn clauses	
$\delta(O)$	$= q_{even}$	$q_{even}(O)$	
$\delta(S, q_{even})$	$= q_{odd}$	$q_{odd}(S(x))$	$\Leftarrow q_{even}(x)$
$\delta(S, q_{odd})$	$= q_{even}$	$q_{even}(S(x))$	$\Leftarrow q_{odd}(x)$
$\delta(+, q_{even}, q_{even})$	$= q_{even}$	$q_{even}(+(x, y))$	$\Leftarrow q_{even}(x), q_{even}(y)$
$\delta(+, q_{even}, q_{odd})$	$= q_{odd}$	$q_{odd}(+(x, y))$	$\Leftarrow q_{even}(x), q_{odd}(y)$

$\ldots$

A uniform framework for

– describing various extensions of ordinary automata (e.g. alternating, two-way automata)

– dealing with arbitrary equational theories

Automata queries as Horn clauses

To test membership of term m at state q we add a clause

$$\perp \Leftarrow q(m)$$

and check whether $\perp$ can be derived from the clauses.

To test non-emptiness of state q we add clause

$$\perp \Leftarrow q(x)$$

To test intersection-non-emptiness of states q_1 and q_2 we add clause

$$\perp \Leftarrow q_1(x), q_2(x)$$

Tree automata and cryptographic protocols

Terms represent messages involved in a protocol

Set of messages known to intruder is expressed by a tree automaton

$$
\begin{array}{rll}
I_C(\texttt{encrypt}(m,k)) & \Leftarrow I_C(m), I_C(k) & \text{Intruder can encrypt messages} \\
I_C(\texttt{pair}(x,y)) & \Leftarrow I_C(x), I_C(y) & \text{Intruder can form pairs} \\
I_{C_{new}}(x) & \Leftarrow I_{C_{old}}(x) & \text{Intruder remembers past messages}
\end{array}
$$

Need for two-way tree automata

New clauses needed for modeling cryptographic protocols:

$I_C(m)$	$\Leftarrow I_C(\mathtt{encrypt}(m,k)), I_C(k)$	Intruder can decrypt messages
$I_C(x)$	$\Leftarrow I_C(\mathtt{pair}(x,y))$	Intruder can unpair messages
$I_C(y)$	$\Leftarrow I_C(\mathtt{pair}(x,y))$	

These clauses destruct terms instead of constructing terms

$\Rightarrow$ Extend one-way tree automata to two-way tree automata

Sometimes we also need alternation clauses: $P(x) \Leftarrow P_1(x), P_2(x)$

Ordinary automata are not expressive enough

Given a regular language L, is the ACU-closure of L regular ?

$$ACU(L) = \{t \mid \exists s \in L \cdot s =_{ACU} t\}$$

Ordinary automata are not expressive enough

Given a regular language L, is the ACU-closure of L regular ?

$$ACU(L) = \{t \mid \exists s \in L \cdot s =_{ACU} t\}$$

No. The set of terms of the form

$$(\ldots((a{+}b){+}a{+}b)\ldots{+}a{+}b)$$

is regular. Its closure is the set of terms with equal number of occurrences of a and b, which is not regular.

Ordinary automata are not expressive enough

Given a regular language L, is the ACU-closure of L regular ?

$$ACU(L) = \{t \mid \exists s \in L \cdot s =_{ACU} t\}$$

No. The set of terms of the form

$$(\ldots((a+b)+a+b)\ldots+a+b)$$

is regular. Its closure is the set of terms with equal number of occurrences of a and b, which is not regular.

Solution: interpret the $+$ operation as a special operation satisfying some equational properties.

Example Consider clauses

$q_1(a)$

$q_2(a)$

$q_3(0)$

$q_4(x+y) \Leftarrow q_1(x), q_2(y)$

$q_5(x) \Leftarrow q_3(x), q_4(x)$

In the absence of equational theories, nothing is accepted at q_5.

Example   Consider clauses

$q_1(a)$

$q_2(a)$

$q_3(0)$

$q_4(x+y) \Leftarrow q_1(x), q_2(y)$

$q_5(x) \Leftarrow q_3(x), q_4(x)$

In the absence of equational theories, nothing is accepted at q_5.

In presence of the equational theory XOR:

$a+a$ is accepted at q_4.

Hence 0 is accepted at q_4.

Hence 0 is accepted at q_5.

Modeling of group key agreement protocol (1)

For each configuration C:

$k_C(e(0))$	intruder knows α
$k_C(e(x+y)) \Leftarrow k_C(e(x)), k_C(y)$	intruder can exponentiate
$k_C(nil)$	intruder knows empty list
$k_C(cons(x,y)) \Leftarrow k_C(x), k_C(y)$	intruder can build lists
$k_C(x) \Leftarrow k_C(cons(x,y))$	intruder can read heads
$k_C(y) \Leftarrow k_C(cons(x,y))$	intruder can read tails

Modeling of group key agreement protocols (2)

Second step:

B expects a message of the form $x; \alpha^y$

B sends the message $\alpha^{N_b}; \alpha^y; \alpha^{y \cdot N_b}$

translated to clauses:

$k_{C_2}(e(Nb); e(y); e(y+Nb)) \Leftarrow k_{C_1}(x; e(y))$

$k_{C_2}(x) \Leftarrow k_{C_1}(x)$

Modeling of group key agreement protocols (4)

Secrecy requirement on A's view of the group key:

$$\perp \Leftarrow k_{C_3}(e(x); y), k_{C_3}(e(x+Na))$$

Translates to intersection emptiness problem of two-way AC automata (decidable)

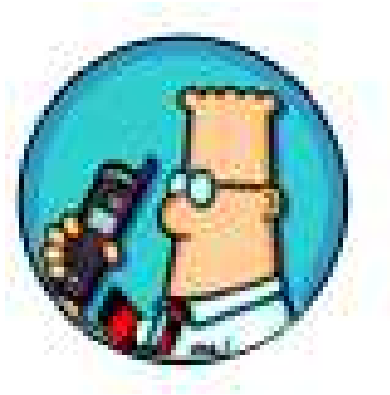
Modeling of the protocol using XOR

Alice

$N_a + K_{ab}$ $\longrightarrow$

$N_b + N_a$ $\longleftarrow$

$S_{ab} + N_b$ $\longrightarrow$



Bob

Translation of the second rule:

$I_C(x+K_{ab}+N_b) \Leftarrow I_{C'}(x)$

Connections with sets of vectors of integers

Consider constants a, b and symbol $+$.

The clauses

$$P(a)$$
$$P(x+a+b+b) \Leftarrow P(x)$$

with final state P define the language

$$\{na + mb \mid n > 0 \wedge m = 2n - 2\}$$

The Parikh image is the set

$$\{(n, m) \mid n > 0 \wedge m = 2n - 2\}$$

The formula involved is a Presburger formula:
formulas built using variables, 0, 1, $+$, logical connectives and quantifiers,
but no multiplication.

A base $\nu \in \mathbb{N}^p$ and periods $\nu_1, \ldots, \nu_k \in \mathbb{N}^p$ define a linear set

$$\{\nu + x_1\nu_1 + \ldots + x_p\nu_p \mid x_1, \ldots, x_p \in \mathbb{N}\}$$

Semilinear sets $\equiv$ finite union of linear sets $\equiv$ Presburger-definable sets

Closed under union, intersection, complementation and projection.

A base $\nu \in \mathbb{N}^p$ and periods $\nu_1, \ldots, \nu_k \in \mathbb{N}^p$ define a linear set

$$\{\nu + x_1 \nu_1 + \ldots + x_p \nu_p \mid x_1, \ldots, x_p \in \mathbb{N}\}$$

Semilinear sets $\equiv$ finite union of linear sets $\equiv$ Presburger-definable sets

Closed under union, intersection, complementation and projection.

The previous example

$$\{(n, m) \mid n > 0 \wedge m = 2n - 2\}$$

is described using base $(1, 0)$ and period $(1, 2)$

This is also the Parikh image of the regular string language $a(abb)^*$.

A base $\nu \in \mathbb{N}^p$ and periods $\nu_1, \ldots, \nu_k \in \mathbb{N}^p$ define a linear set

$$\{\nu + x_1\nu_1 + \ldots + x_p\nu_p \mid x_1, \ldots, x_p \in \mathbb{N}\}$$

Semilinear sets $\equiv$ finite union of linear sets $\equiv$ Presburger-definable sets

Closed under union, intersection, complementation and projection.

The previous example

$$\{(n, m) \mid n > 0 \wedge m = 2n - 2\}$$

is described using base $(1, 0)$ and period $(1, 2)$

This is also the Parikh image of the regular string language $a(abb)^*$.

Parikh's Theorem: The Parikh image of a regular string language is semilinear,

A base $\nu \in \mathbb{N}^p$ and periods $\nu_1, \ldots, \nu_k \in \mathbb{N}^p$ define a linear set

$$\{\nu + x_1\nu_1 + \ldots + x_p\nu_p \mid x_1, \ldots, x_p \in \mathbb{N}\}$$

Semilinear sets $\equiv$ finite union of linear sets $\equiv$ Presburger-definable sets

Closed under union, intersection, complementation and projection.

The previous example

$$\{(n, m) \mid n > 0 \wedge m = 2n - 2\}$$

is described using base $(1, 0)$ and period $(1, 2)$

This is also the Parikh image of the regular string language $a(abb)^*$.

Parikh's Theorem: The Parikh image of a regular string language is semilinear,

and also the Parikh image of a context-free string language is semilinear.

Consider clauses

$$q(5a)$$

$$q(x+y+z) \Leftarrow q(x), q(y), q(z)$$

q accepts the language $\{na \mid n = 5 \vee \exists m \cdot n = 15 + 10m\}$.

Consider clauses

$$q(5a)$$
$$q(x+y+z) \Leftarrow q(x), q(y), q(z)$$

q accepts the language $\{na \mid n = 5 \vee \exists m \cdot n = 15 + 10m\}$.

This can also be represented by the context-free language defined by the grammar

$$q \longrightarrow aaaaa$$
$$q \longrightarrow qqq$$

Consider clauses

$$q(5a)$$
$$q(x+y+z) \Leftarrow q(x), q(y), q(z)$$

q accepts the language $\{na \mid n = 5 \vee \exists m \cdot n = 15 + 10m\}$.

This can also be represented by the context-free language defined by the grammar

$$q \rightarrow aaaaa$$
$$q \rightarrow qqq$$

$\Rightarrow$ If we consider clauses corresponding to ordinary (one=way) tree automata (containing $+$ and other symbols), them modulo theories ACU, XOR and Abelian Groups, the languages are closed under intersection and emptiness is decidable.

Complementation

Consider languages modulo XOR:

$L_1 = \{f^m(a) + f^n(a) \mid m, n \geq 0\}$

$L_2 = \{0\}$

$L_1 \setminus L_2 = \{f^m(a) + f^n(a) \mid m, n \geq 0 \wedge m \neq n\}$

L_1, L_2 accepted by one-way XOR automata, but not $L_1 \setminus L_2$.

$\Rightarrow$ One-way XOR automata not closed under complementation

Counter-example exists also for the Abelian Groups theory.

For ACU theory, we have closure under complementation.

Elimination of two-wayness

Example With theory XOR, given clauses

$$q(x) \Leftarrow p(f(x))$$
$$p(x + y + z) \Leftarrow p_1(x), p_2(y), p_3(z)$$
$$p_1(f(x)) \Leftarrow q_1(x)$$
$$p_2(a)$$
$$p_3(a)$$

we deduce clause

$$q(x) \Leftarrow q_1(x)$$

Elimination of two-wayness

Example With theory XOR, given clauses

$$q(x) \Leftarrow p(f(x))$$
$$p(x+y+z) \Leftarrow p_1(x), p_2(y), p_3(z)$$
$$p_1(f(x)) \Leftarrow q_1(x)$$
$$p_2(a)$$
$$p_3(a)$$

we deduce clause

$$q(x) \Leftarrow q_1(x)$$

In general the second clause may not be present but implied by other clauses.

$\Rightarrow$ Use Presburger-formula to represent the set of all such formulas.

An undecidability result

Alternation clauses: $q(x) \Leftarrow q_1(x), q_2(x)$

encode 2 counter automata

$\Rightarrow$ emptiness undecidable for alternating automata
(for theories ACU and Abelian Groups)

For theory XOR we still have decidability.

Other clauses: Petri nets and VASS

Consider clauses of the form

$$q(a{+}2b)$$
$$q(x + 2a + 5b) \Leftarrow q(x)$$
$$q(x) \Leftarrow q(x + 6b)$$

equivalently

$$q(1, 2)$$
$$q(x + (2, 5)) \Leftarrow q(x)$$
$$q(x) \Leftarrow q(x + (0, 6))$$

The last clause can be applied only when $x \geq (0, 6)$.

These clauses can perform subtraction: these define Petri nets or VASS (Vector Addition Systems with States). We can now define non-semilinear sets.

Intersection-emptiness etc. continue to be decidable, but are expensive.

Branching VASS

Suppose we consider subtraction, together with branching addition.

$$q(\nu)$$
$$q(x+\nu) \Leftarrow q_1(x)$$
$$q(x) \Leftarrow q_1(x+\nu)$$
$$q(x+y) \Leftarrow q_1(x), q_2(y)$$

The decidability of reachability (membership, intersection-non-emptiness) is open.

Equivalent to decidability of provability in MELL (Multiplicative Exponential Linear Logic).