Technische Universität Darmstadt
Fachbereich Informatik
Fachgebiet Sicherheit in der Informationstechnik

Anwendungen und Algorithmen basierend auf Prinzipien eines künstlichen Immunsystems

Vortragsthema

Online Negative Databases

Referent: Mohammed Naoufal Adraoui

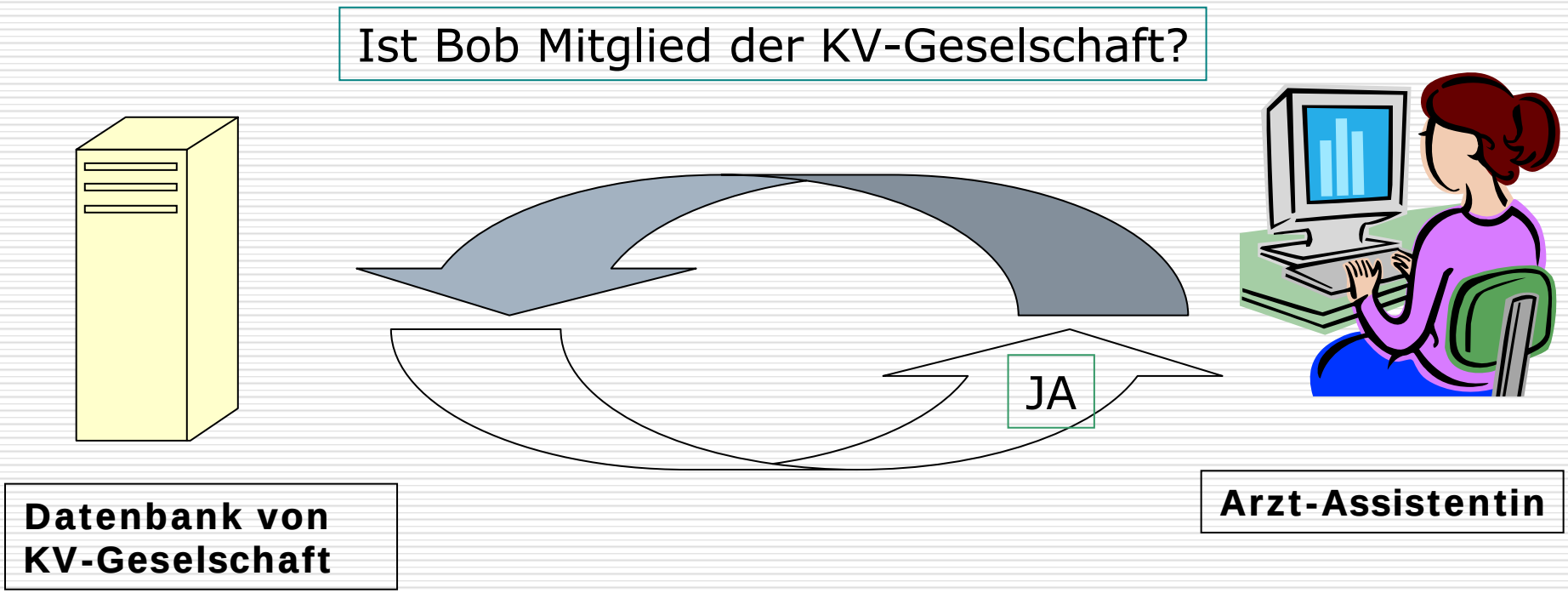
04 Mai 2005

angefertigt bei Prof. Dr. C. Eckert
betreut von Thomas Stibor

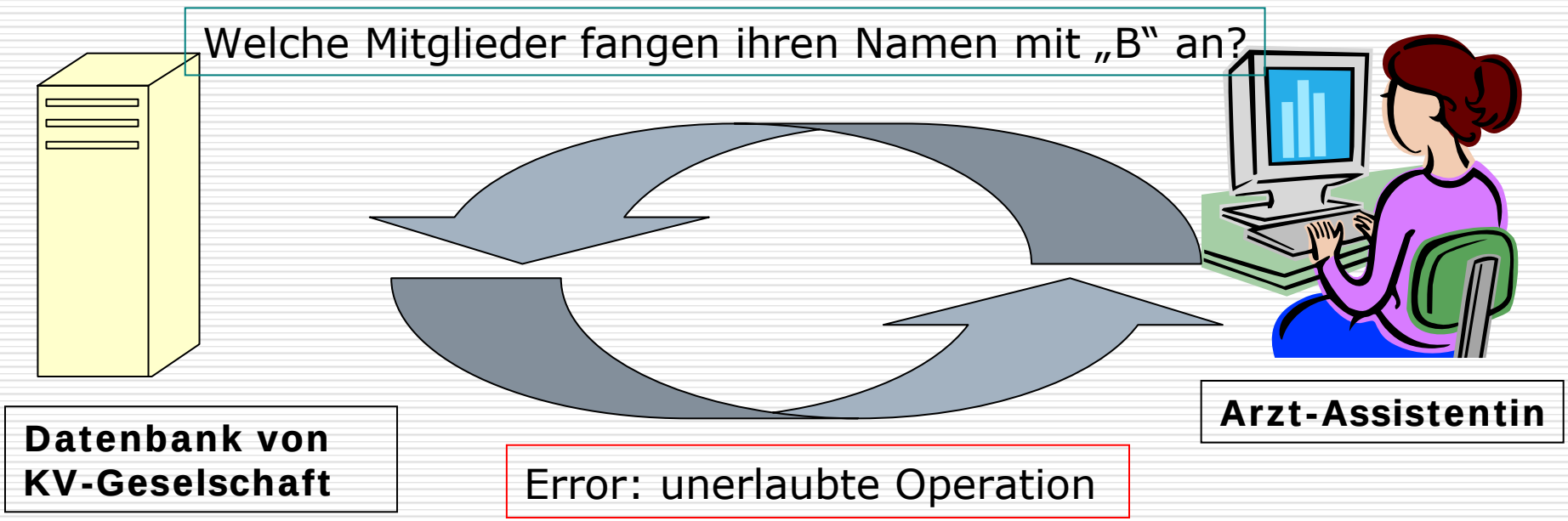
Negative Datenbanken

- Inhalt
  - Ziel und Motivation
  - Biologisches Hintergrund
  - Definition und Begriffe
  - Algorithmen
  - Fazit und Diskussion

1.Motivation



1.Motivation



- Ziel

Implementierung von einer Datenbank, die die Privatheit (Privacy) der Daten garantiert durch Benutzung von Konzepten aus dem Immunsystem

➔ Aha gut, und wie schafft man das?!!

2.Ein wenig Biologie

↳Das Immunsystem muss zw. eigenen Molekülen (self) und körperfremden Molekülen (nonself) unterscheiden

↳Dafür wird eine negative Entdeckung bzw. Selektion verwendet

➔Warum?

↳„ Dadurch lassen sich die Lymphozyten gegen Selbstattacken währen“

➔ was kann man aus diesem Konzept herleiten?!!

3.Negative-detection Schema

- Dieses Schema wurde von mehreren artifiziellen immunen Systemapplikationen (AIS) für Anomalienentdeckung benutzt
- Ähnlich zum immunen System wird die Identifikation von Anomalien durch
  - eine Menge von Strings bzw. Detektoren
  - eine Match-Regel zur Erkennung von Nonself aber kein Self

erkannt.

➔ **Informationen über Self verstecken (Privacy)**

➔ **Wie?**

4.Negative Datenbanken

- Was ist eine negative Datenbank?
  - Menge von Daten wird durch ihr *Komplement* repräsentiert
  - 1.Annahme: $\exists$ U:= $\{0,1\}^n$; n>0
    - Positive Datenbank DB (self) d.h. die Menge von Strings mit Länge n einer DB
    - Die anderen Strings mit Länge n die **nicht** in DB existieren (nonself) d.h. U-DB
  - 2.Annahme:
    - $\forall$ S $\in$ DB : S ist in DB explizit und nicht komprimiert dargestellt
    - $\forall$ S´ $\in$ U-DB: S´ soll komprimiert dargestellt werden => Die komprimierte Form von U-DB wird als NDB bezeichnet
  - ➔Frage: wie und warum komprimiert man die S´?

4.1.Konstruktion von NDB

- Prinzip
  - Strings mit einigen Überlappungen in ihren Darstellungen werden in Untermengen zusammengefasst
  - Diese Überlappungen werden in einer klaren Weise dargestellt
    - Man erweitert das Alphabet durch das Don´t care Symbol * .
    - Die Überlappungen durch * in NDB darstellen unter der Voraussetzung:

      $s \in DB \Rightarrow \forall\, t_i \in NDB \;\; t_i$ DISAGREE_WITH s

      DISAGREE_WITH:= mindestens ist eine Stelle verschieden
  - ➔Geheimhaltung der positiven Daten wird dadurch erreicht!!

4.1.Konstruktion von NDB

- Beispiel

DB	U-DB	NDB
000	001	
010	011	**1
100	101	
	110	11*
	111	

- Frage: Ist es nicht einfach diesen Vorgang in die andere Richtung umzukehren? D.h. NDB=>U-DB=>DB
- Nein. Warum? Das Problem ist isomorphe zu dem NP-schweren Problem SAT.

NDB <=> SAT

- Was ist SAT?
  - Gegeben: eine boolesche Formel F dargestellt in der Konjunktiven Normalform (KNF)
  - Frage: ist F erfüllt?
- SAT und NDB sind equivalent
  - Beispiel

$0 => x_i$; $1 => \neg x_i$

Boolesche Formel F	NDB
$(x_2$ or $\neg x_5)$ and	*0**1
$(\neg x_2$ or $x_3)$and	*10**
$(x_2$ or $\neg x_4$ or $\neg x_5)$and	*0*11
$(x_1$ or $\neg x_3$ or $x_4)$and	0*10*
$(\neg x_1$ or x_2 or $\neg x_4$ or $x_5)$	10*10

$\{x_1$=TRUE, x_2=TRUE, x_3=TRUE, x_4=FALSE, x_5= FALSE$\}$=>S=11100
Also S NOT_REPRESENTED_IN NDB

4.2.Erstellen und warten von NDB

- String matching:
  - Definition: zwei Strings haben ein „Match“ wenn alle ihren Stellen ein Match haben. Das Don´t care Symbol hat ein Match zu jedem Stellenwert
  - Eigenschaft 1: $Y,X \in \{0,1,*\}^1$
    (1) Y SUBSUMED_BY X
    (2) $\forall S \in \{0,1,*\}^1$: S MATCHED_BY Y => S MATCHED_BY X
    (3) X wird durch Ersetzen von einigen definierten Stellen in Y durch * erhalten

    (1)⇔(2)⇔(3)
  - Eigenschaft 2: Eine Menge von 2^n Strings, die bis auf n Stellen gleich sind, entspricht (match) dieselbe Menge von Strings, die durch das Ersetzen dieser n Stellen durch * dargestellt wird.
    - Beispiel

      0000
      0001 } 00**
      0010
      0011

n ist nicht die Länge des Strings

4.2.1. Initialisierung von NDB

- 1ste Möglichkeit
  - Initialisierungswert einer DB ist die leere Menge
  - D.h. NDB= U
  - Also man kann NDB so darstellen $*^{1}$; 1 ist die Länge des Strings
    - ➔Keine gute Lösung. Warum?
    - ➔Jeder kann merken welche Strings, die die NDB darstellt
    - ➔Es muss schwer sein, den Inhalt von DB festzustellen auch wenn DB leer ist.
- 2te Möglichkeit
  - Einsatz von einem Algorithmus, der prinzipiell die Klauseln einer unerfüllten SAT Formeln zur Einträge der NDB umwandelt
  - Prinzip:
    - Selektieren von m Bitstellen
    - Kreieren für jede mögliche Bitbelegung V_p dieser Stellen ein String mit V_p und * in der anderen Stellen.
    - NDB hat dabei 2^m Felder, die 2^{l-m} unterschiedlichen Strings entsprechen

4.2.1. Initialisierung von NDB

- 2te Möglichkeit (cont.)
  - Algorithmus:

Empty_NDB_Create(l)

1. Pick ⌈log(l)⌉ bit positions at random
2. For every possible assignment V_p of this positions{
3. select k_1 randomly $1 \leq k_1 \leq l$
4. for j=1 to k_1 {
5. select an additional *n* distinct p
6. for every possible assignment V_q of these positions{
7. Pick k_2 bits at random from $V_p.V_q$.
8. Create a entry for NDB with the k_2 chosen bits and fill the remaining $l-k_2$ positions with the don´t care symbol}}}

mehr als ein String, das einen Muster entspricht, hinzufügen

Das originale Muster mit n anderen Stellen erweitern (n=3 =>3-SAT)

NDB-Eintrag kreieren durch Benutzung von einer Untermenge von Stellen der gewählten Mustern

4.2.2.Aktualisierung von NDB

- „insert“↔ „delete“
  - „insert into DB“ ⇔ „delete from DB“
  - Prinzip:
    - String x und aktuelle NDB als Input nehmen
    - String y ausgeben, das x einordnet ohne einen anderen String in DB zu entsprechen
  - Algorithmus:

Negative_Pattern_Generate(x,NDB)

1. Create a random permutation π
2. For all specified bits b_i in $\pi(x)$
3. Let x´ be the same as $\pi(x)$ but with b_i flipped
4. if x´ is subsumed by some string in π(NDB)
5. $\pi(x) \leftarrow \pi(x)$ – i^{th} bit (set value to *)
6. Keep track of the i^{th} bit in a set indicator vector(SIV)
7. Randomly choose $0 \le t \le |SIV|$
8. R ←t randomly selected bits from SIV
9. Create a pattern V_k using $\pi(x)$ and the bits indicated by R.
10. Return $\pi'(V_k)$

x ist über{0,1,*} definiert

Die inverse Permutation von π

4.2.2.Aktualisierung von NDB

- Insert into NDB
  - Eine Untermenge von Strings in NDB hinzufügen
  - Die Irreversibilitätseigenschaft von NDB beibehalten
  - Algorithmus:
    - **Insert (x,NDB)**

1. Randomly choose $1 \leq j \leq l$
2. for k= 1 to j do

Mehrere darstellende NDB Einträge von x kreieren

3. Randomly select from x at most n distinct unspecified bit positions
4. For every possible bit assignment B_p of the selected positions
5. $x' \leftarrow x.B_p$
6. $y \leftarrow$ Negative_Pattern_Generate (x´, NDB)
7. add y to NDB

4.2.2.Aktualisierung von NDB

- Delete from NDB
  - Löschen eines Strings X aus der NDB. X ist über {0,1,*} definiert.
  - Keine leichte Sache da mehrere NDB Einträge dasselbe String gleichzeitig darstellen können;
  - Und ein NDB Eintrag kann mehrere Strings darstellen, die man nicht alle löschen möchte.
  - Algorithmus:
    - **Delete(x,NDB)**

1. Let D_x be all the strings in NDB that match x
2. Remove D_x from NDB.
3. For all $y \in D_x$
4. for each unspecified position q_i of y
5. Create a new string y´using the specified bits of y and the complement of the bit specified at the i^{th} position of x.
6. Insert(y´, NDB)

4.2.2.Aktualisierung von NDB

- Delete from NDB (cont.)
  - Beispiel

x	D_x	All but x
101001	1*1*0*	111*0* 1*110* 1*1*00

  - Problem: |NDB| kann dadurch ins Ufer wachsen!!
  - In einer Implementierung muss es verhindert werden, dass |Dx| = f(|NDB|)
  - Und/oder garantieren dass |NDB|≤ Konstante

4.2.2.Aktualisierung von NDB

- Zusammenfassendes Beispiel

Empty_NDB_Create(4)	Delete(1111,NDB)	Insert(1111,NDB)
000*	000*	000*
001*	001*	001*
01*0	01*0	01*0
01*1	01*1	01*1
10*1	10*0	10*0
111*	10*1	10*1
110*	110*	110*
	11*0	11*0
	*110	*110
		11

- Ein Demoprogramm findet man unter:
http://esa.ackleyshack.com/ndb/php/incremental/negdb-main.php

5.Fazit

- Negative Datenbanken basieren sich auf dem Konzept der komplementären Darstellung
- Für eine gegebene NDB ist es schwer den Inhalt von DB herauszufinden => Privacy
- Erfolgreiche Unterscheidung zw. Self und Nonself
- Optimale Anzahl von den angeforderten Detektoren

Literaturverzeichnis

Schriftliche Dokumente:

- F. Esponda,S. Forrest, and P. Helman. Enhancing privacy through negative representations of data. Technical report, University of New Mexico,2004
- F. Esponda, E.S. Ackley,S. Forrest, and P. Helman. Online negative databses. Technical report, University of New Mexico
- J. Johannsen, Algorithmen für das SAT Problem,Skript zur Vorlesung. Lehr- und Forschungseinheit für Theoretische Informatik,Institut für Informatik der Ludwig-Maximilians-Universität München , 2002/2003.
- L. N. de Castro, F. J. Von Zuben. Artificial immune systems; Basic theory and applications. Technical report, December, 1999
- L. N. de Castro, F. J. Von Zuben. Artificial immune systems; A survey of applications. Technical report, February, 2000

Links:

- **http://esa.ackleyshack.com/ndb/php/incremental/negdb-main.php**

KEINE FRAGEN?