„Gedanken zum Fundament des Informatikturms“
Jürg Nievergelt, ETH Zürich, 1995

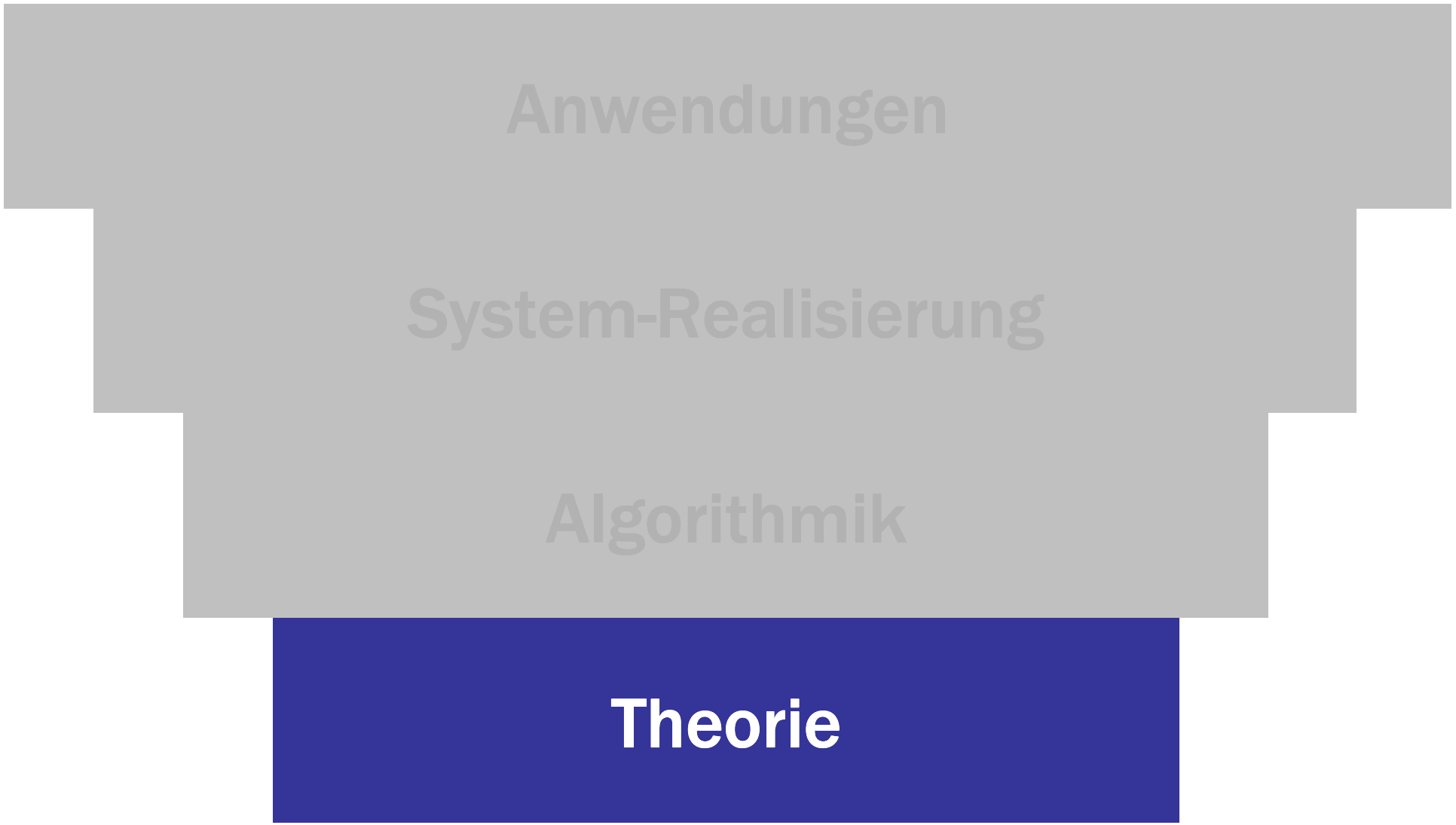
„Gedanken zum Fundament des Informatikturms“
Jürg Nievergelt, ETH Zürich, 1995

Anwendungen: das „Gesicht“ der Informatik
Marketing & Verkauf; Installation & Betrieb

System-Realisierung: Projekte
Programmierung, Projektleitung

Algorithmik: klein, aber fein!
Forschung & Entwicklung

Theorie: was ewig währt
Die Grübler


Anwendungen
System-Realisierung
Algorithmik
Theorie

Warum wir nur einen kurzen Abstecher in die theoretische Informatik machen ...

I would advise students to pay more attention to the fundamental ideas rather than the latest technology. The technology will be out-of-date before they graduate. Fundamental ideas never get out of date.

However, what worries me about what I just said is that some people would think of Turing machines and Goedel's theorem as fundamentals.

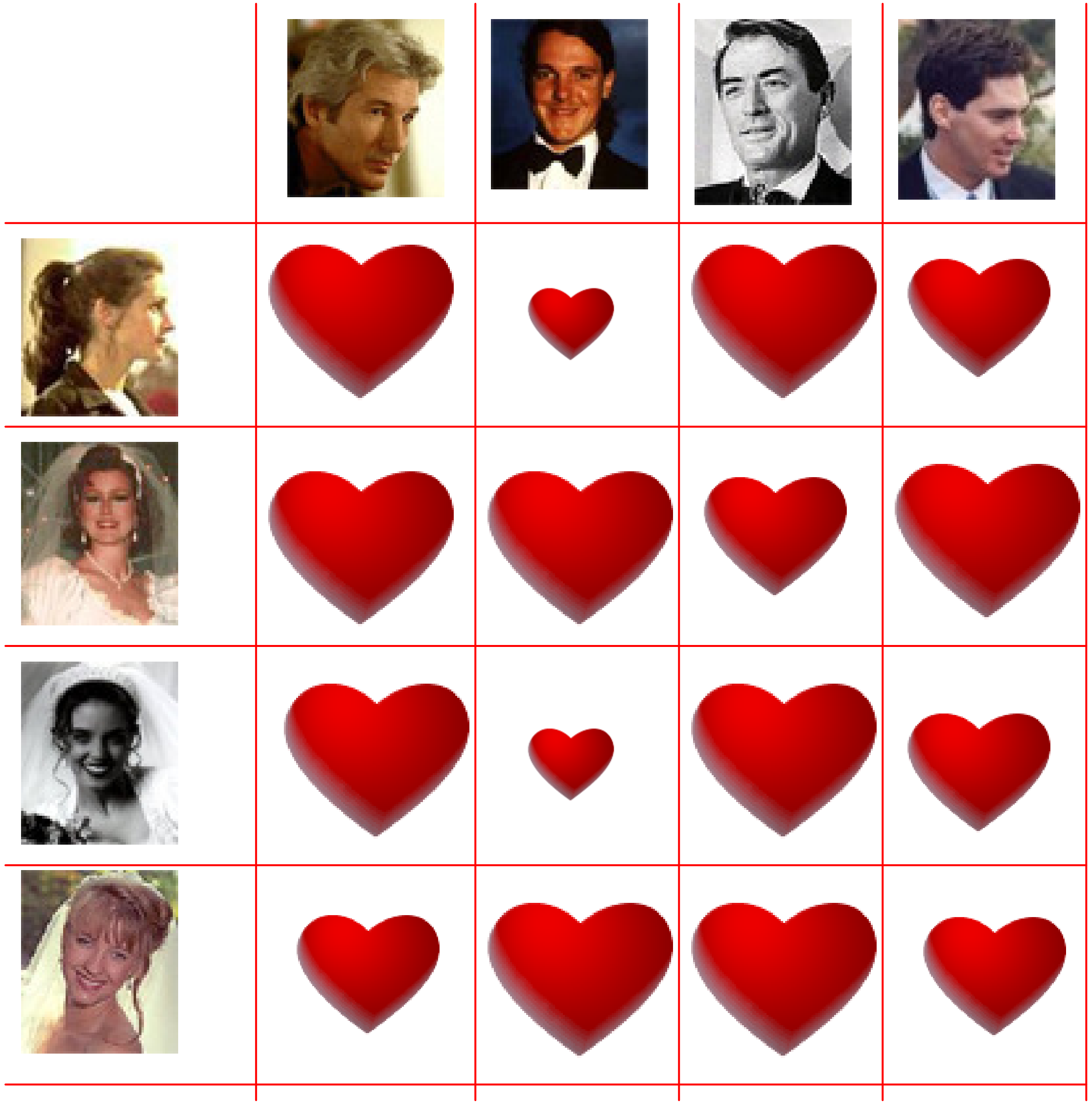
I think those things are fundamental but they are also nearly irrelevant. I think there are fundamental design principles, for example structured programming principles, the good ideas in "Object Oriented" programming, etc.

David Parnas (1999). Software Engineering Notes, Vol. 24, No. 3.
http://www.sigsoft.org/SEN/parnas.html.

Beispiel: Wie sicher ist mein Geld ... ?

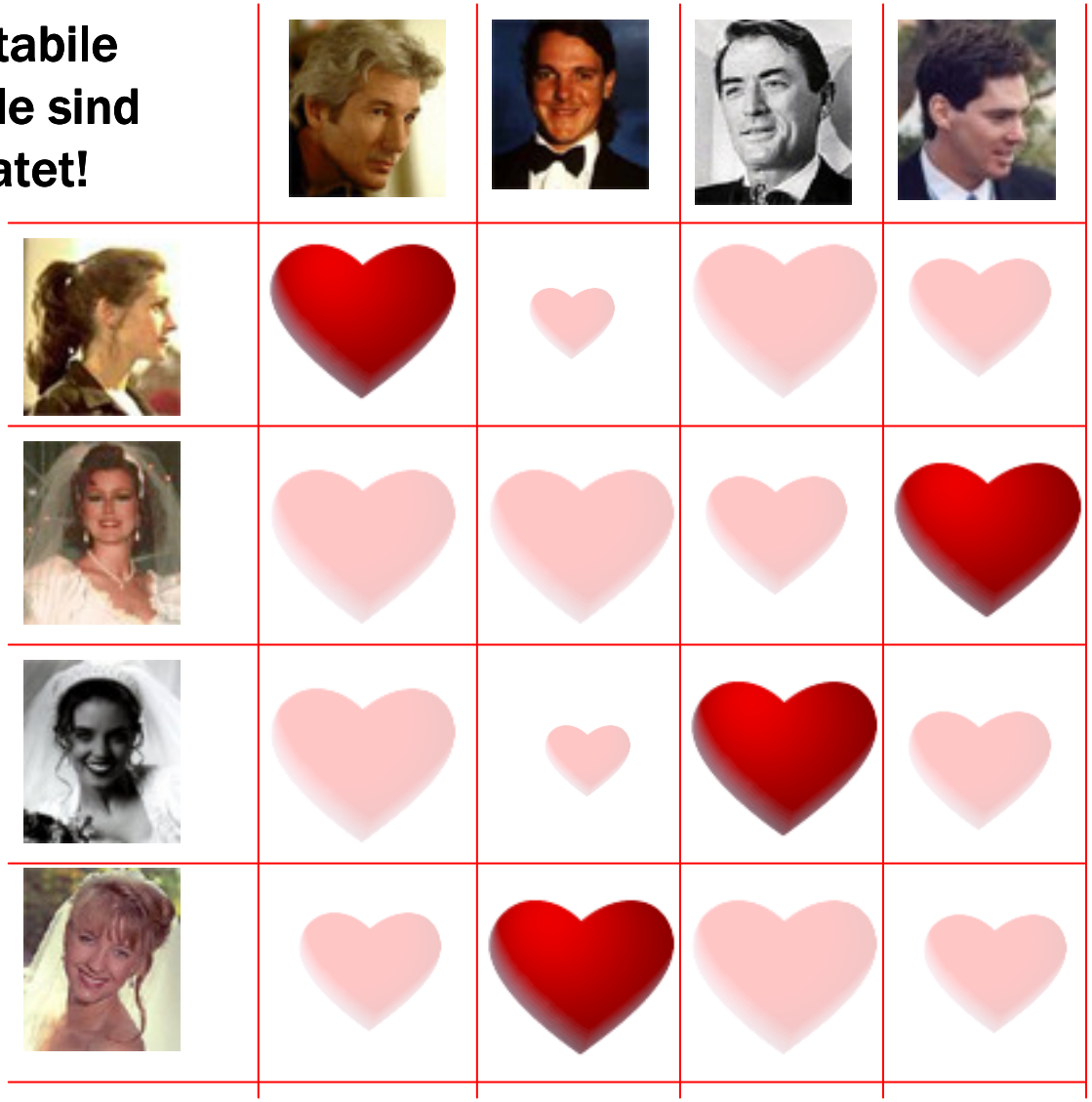
Oder anders gefragt:
wie schwierig ist es, die Verschlüsselung zu knacken?

Beispiel: Willst Du mich heiraten?



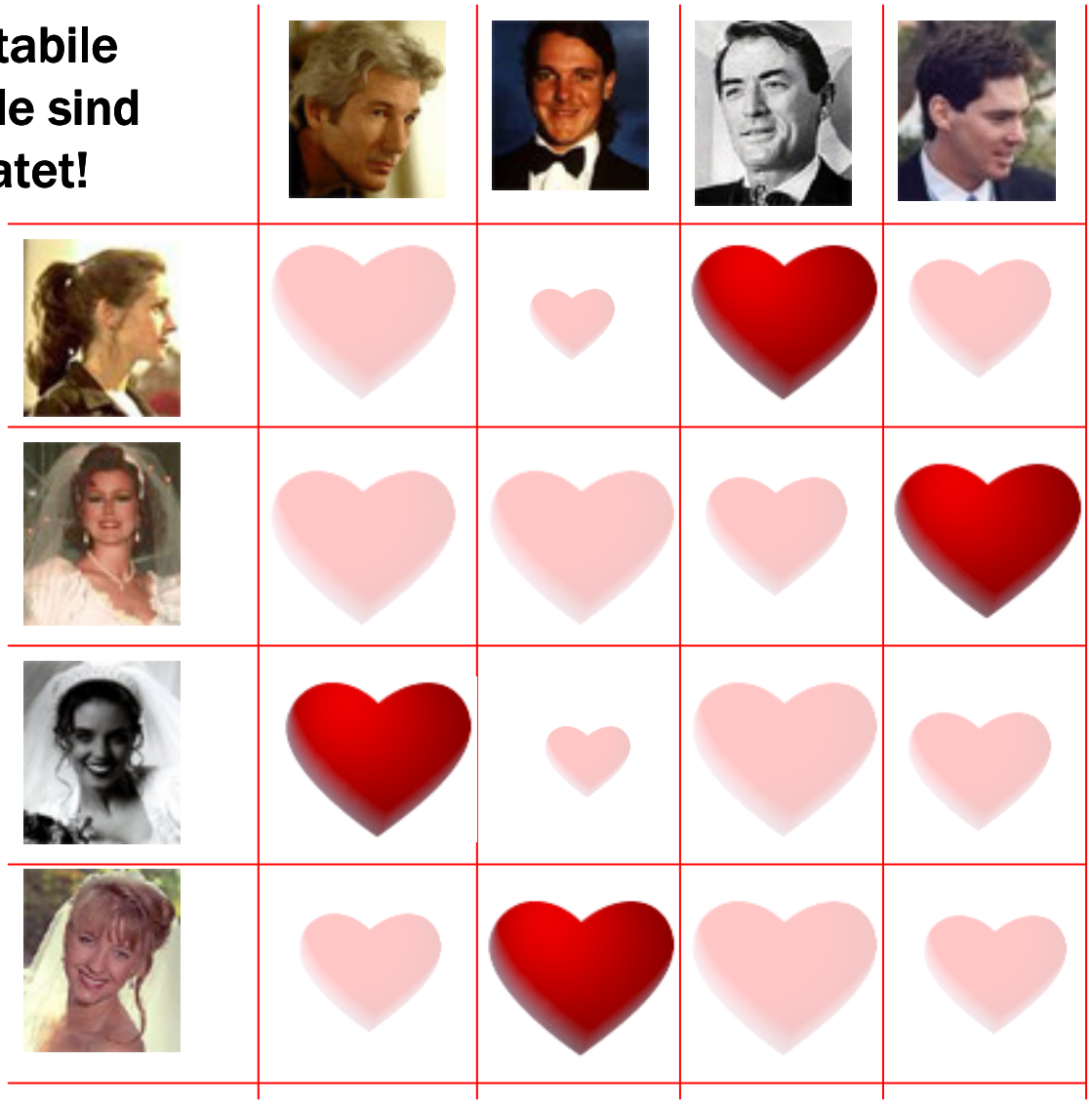
… und wenn sie nicht gestorben sind, so leben sie …

Eine mögliche „stabile Verheiratung“: alle sind glücklich verheiratet!

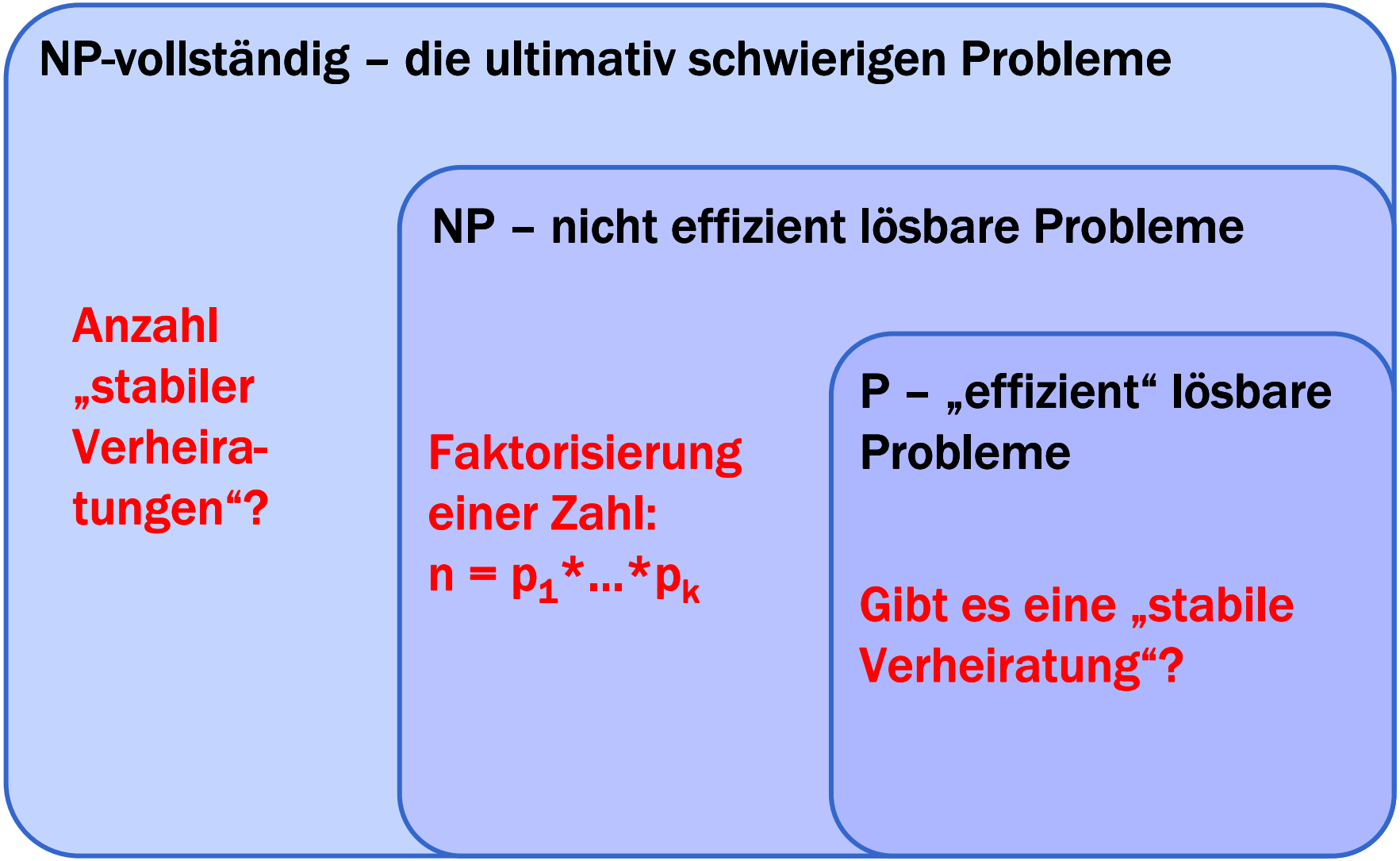


… noch heute …

Eine mögliche „stabile Verheiratung“: alle sind glücklich verheiratet!



Probleme, Probleme!



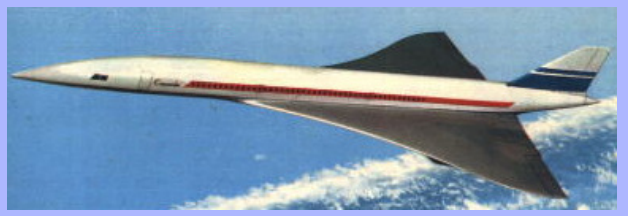
Von Igeln und von Hasen – was heisst hier „effizient“?

NP – nicht effizient lösbare Probleme

Faktorisierung einer Zahl:
$n = p_1 * \ldots * p_k$

P – „effizient“ lösbare Probleme

Gibt es eine „stabile Verheiratung“?



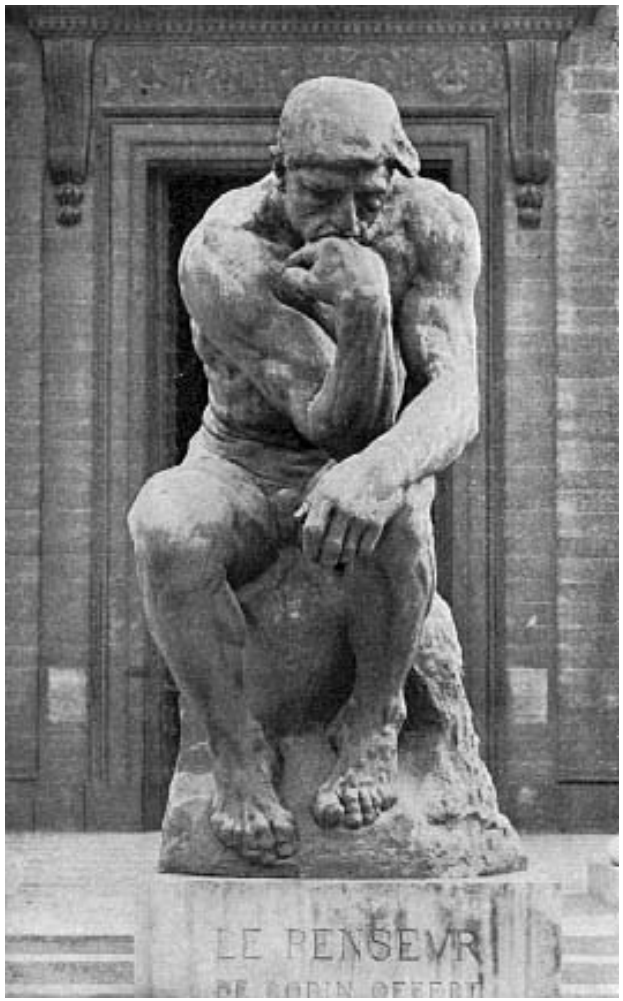
Die 1‘000‘000 US$ Frage

P = NP oder P # NP ?

Eigenschaft von NP-vollständigen Problemen:

wenn jemand für eines davon eine „effiziente“ Lösung findet, wäre das der Beweis dafür, dass alle NP-Probleme effizient gelöst werden können: P = NP!

Theorie: Was tun Informatiker?



Auguste Rodin, „Der Denker“

Sie denken.

Sie grübeln.

Sie beweisen.

Vielleicht.

Sie veröffentlichen.

In obskuren Zeitschriften.

Grundsatzfragen der Informatik

Welche Probleme können mit welchen Mitteln wie effizient gelöst werden?

Welche Probleme können mit gegebenen Mitteln überhaupt gelöst werden ?

abhängig vom „Programmiermodell“:

- **Automaten**
- **Turingmaschinen**
- **Quantencomputer**
- **Bio-Computer ?**
- **?**

Berechnungsmodell – oder: Was kann Kara, was kann er nicht?

Abhängig von Annahme:

1. **„Read-Only World“:**
   Kara darf die Welt nicht verändern

   ⇒ **Endlicher Automat**

2. **„In-Place Memory World“:**
   Kara darf einen beschränkten Bereich der Welt verändern

   ⇒ **Spezialfall Linear Bounded Automaton**

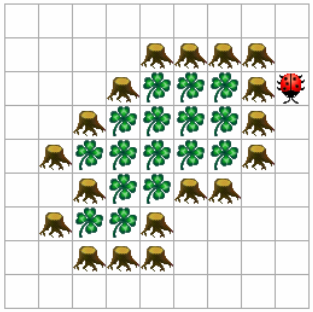
3. **„Read-Write World:**
   Kara darf die Welt beliebig verändern

   ⇒ **Turing-Maschine** (bei unbeschränkter Weltgrösse)

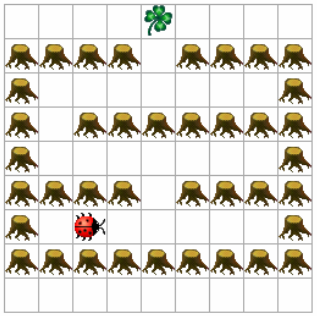
… und trotzdem kann er einige interessante Aufgaben lösen!

Read-Only World

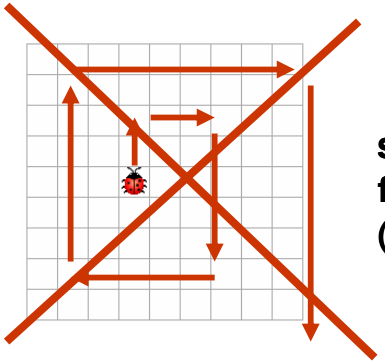
Varianten von „Wand entlang“:



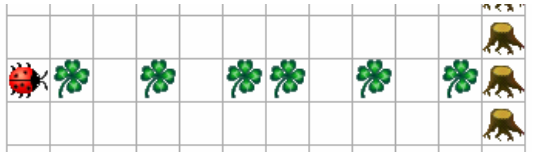
Wand entlang laufen



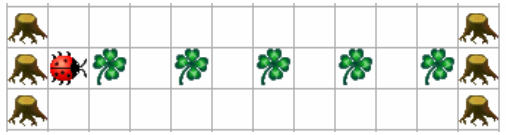
Labyrinthe lösen



sich auf Spiral-Weg fortbewegen (in leerer Welt)



kommt  in der Zeile vor?

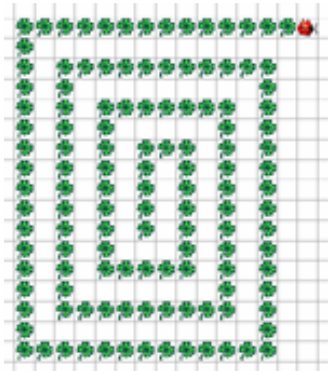


besteht die Zeile aus 

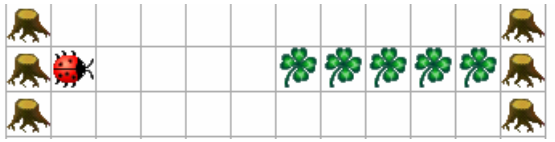
besteht die Zeile aus *n* gefolgt von *n*

… und trotzdem kann er einige interessante Aufgaben lösen!

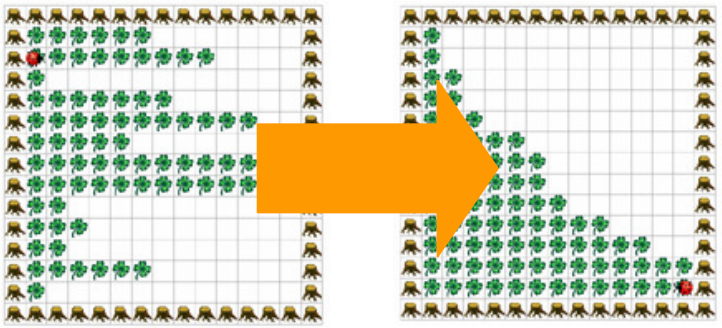
In-Place Memory World



Spirale zeichnen



besteht die Zeile
aus *n*
gefolgt von *n*

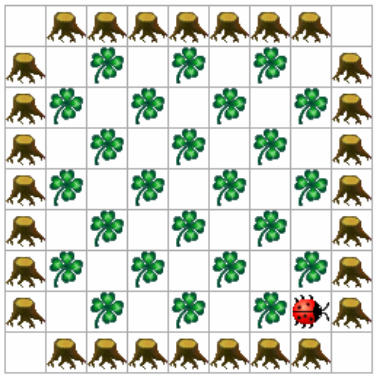


Kleeblattbalken sortieren

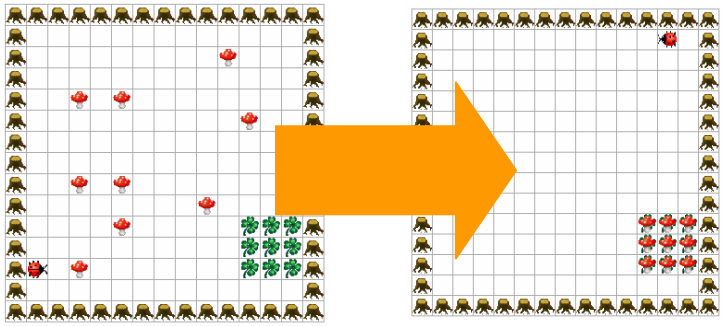
Read-Only World

… und trotzdem kann er einige interessante Aufgaben lösen!

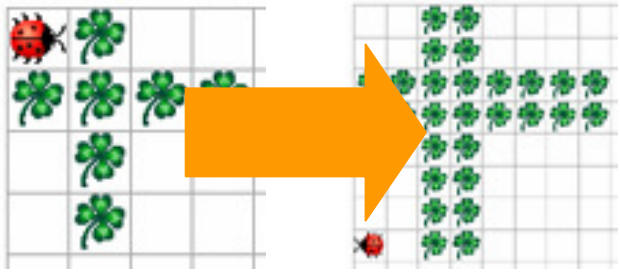
In-Place Memory World



Schachbrett-
muster legen



„Sokoban“ spielen

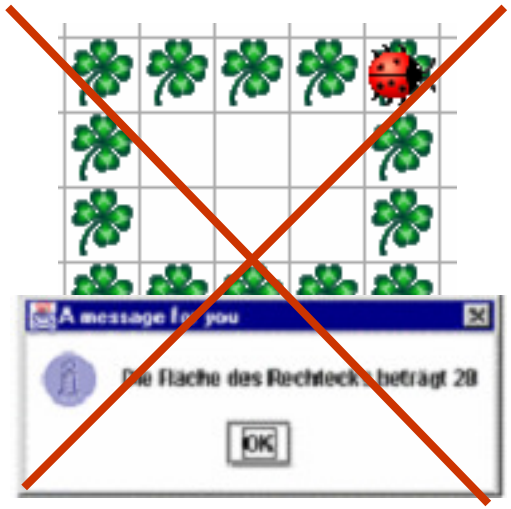


Kleeblattbild skalieren

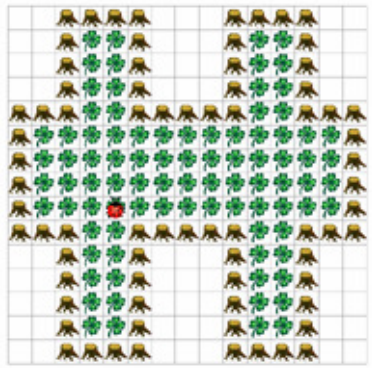
Read-Only World

… aber es gibt auch unlösbare Aufgaben!

Read-Write Memory World



Fläche eines beliebig grossen Rechtecks berechnen

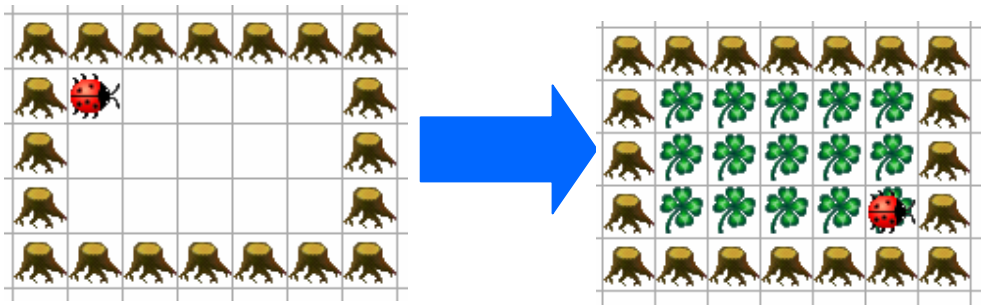


beliebigen begrenzten „Raum“ „parkettieren“

In-Place Memory World

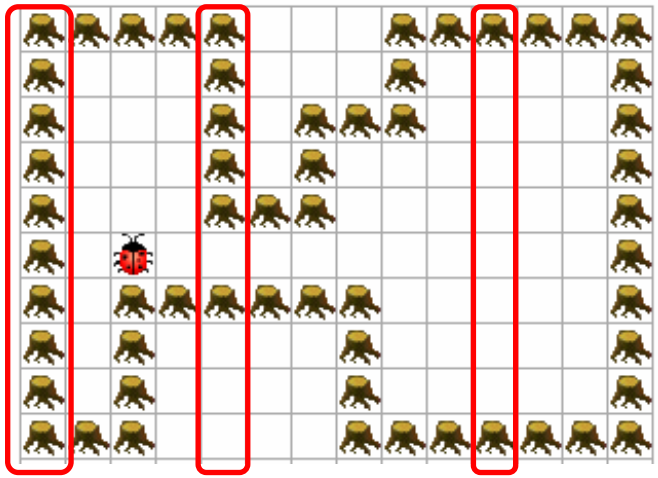
Read-Only World

Beispiel: Raum „parkettieren“



Rechteckige Räume:

✓ **kein Problem (2 Zustände)**

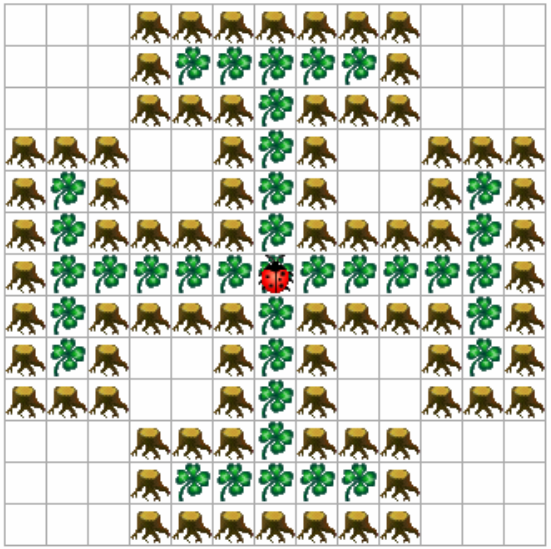


„Vertikal rechteckig konvexer“ Raum:

- **in jeder Spalte höchstens zwei Wände**

✓ **kein Problem (3 Zustände)**

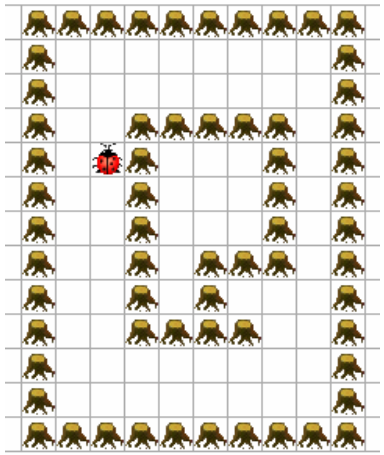
Beispiel: Raum „parkettieren“



beliebig geformte Räume:

☹ **es scheint intuitiv unmöglich zu sein...**

gibt es dafür einen Beweis?



... und was ist mit Hindernissen in den Räumen?

Kara kann es eben doch!

Den Beweis dafür liefert:
Horst Müller: A One-Symbol Printing Automaton Escaping from every labyrinth. Computing 19, 95-110 (1977).

12. März 2001

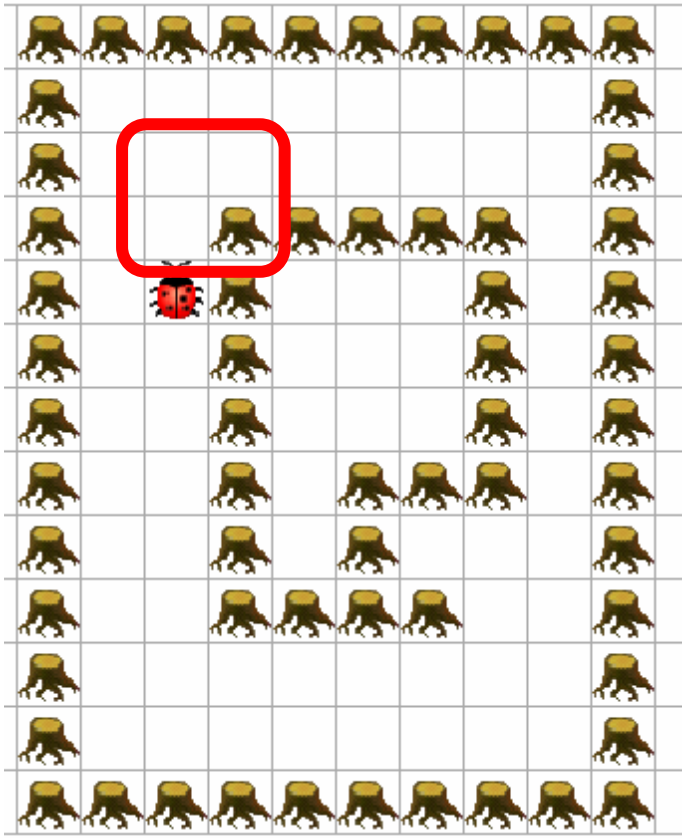
Kara als Labyrinth-Bewältiger

Horst Müller

Über die Konstruktion eines Kara-Käfers,
der einen geschlossenen, endlichen
zusammenhängenden Raum mit Blättern
pflastert (parkettiert, voll absucht)

(Mehr Information auf der Webseite zu Kara.)

Die Kernidee des Algorithmus



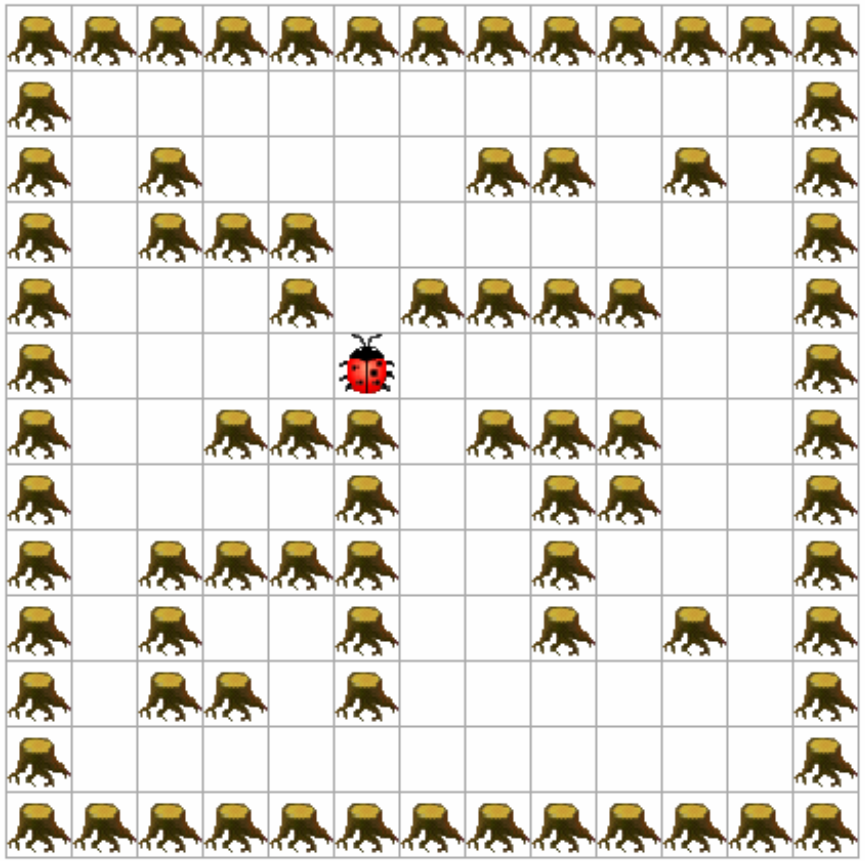
(Stark vereinfacht!)

Man will Backtracking implementieren. Eigentlich müsste man dazu auf ein Feld einen „Pfeil“ legen können: „von daher ist Kara gekommen“.

Das wird simuliert, indem vier Felder zu einem „Makrofeld“ zusammengefasst werden. Dann überlegt man sich die Kombinationen von diesen Feldern mit Kara, Kleeblättern und Bäumen – eine clevere und nicht-triviale Auswahl dieser Kombinationen implementiert man dann als „Pfeile“. Und schon kann man Backtracking implementieren...

Hinweis: ein JavaKara-Programm, das das Verfahren umsetzt, ist rund 20 Seiten lang...

Beispiel: Raum „parkettieren“



beliebig geformte Räume, mit beliebig geformten Hindernissen:

☺ **und es geht doch – wenn Kara „würfeln“ darf…**

Kara kann auch nicht-deterministische Automaten ausführen…