ULF NILSSON, ULFNI@IDA.LIU.SE

Typografi

Typografins syften...

Att förmedla ett innehåll

Att antyda innehållets
karaktär och vikt

Estetiska kvaliteter

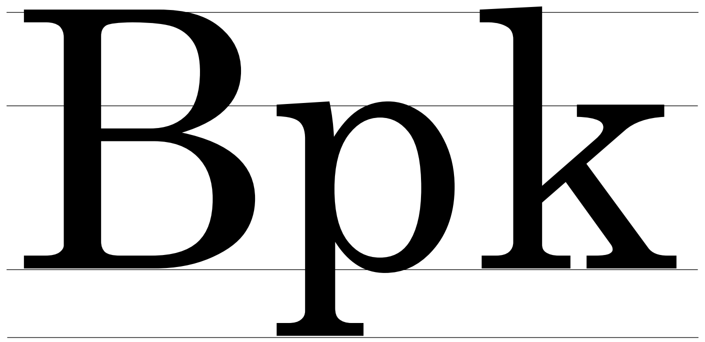
Typografisk kunskap består av...

…fakta, hantverk och tyckande.

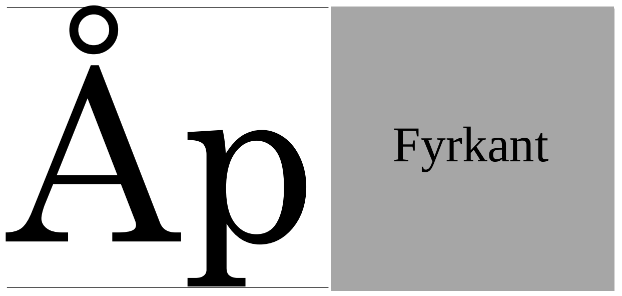
Vad du än gör, ...

GÖR DET KONSEKVENT!!!

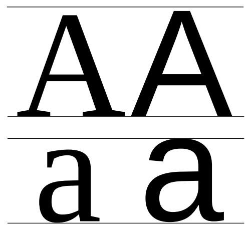
Teckensnitt



Teckenstorlek



Teckenstorlek (grad)



128 p. Times och Helvetica

Antikva

Times
ABCabc

Perpetua
ABCabc

Baskerville
ABCabc

Palatino
ABCabc

Century schoolbook
ABCabc

San Serif / Grotesk / Linjär

Futura
ABCabc

Gill Sans
ABCabc

Helvetica
ABCabc

Arial black
ABCabc

Frutiger
ABCabc

(O)proportionerliga teckensnitt

Helvetica
ABCabc

Times Roman
ABCabc

`Courier`
`ABCabc`

Användning teckensnitt

Antikva i brödtext

Linjärer kan användas i
rubriker och bilder

Programkod i oproportionell

Kursiv

En svart fjäder

En svart fjäder

Bokstäver

Information

Information

jag har hvdvrk

Diverse

VERSALER
KAPITÄLER

abc 1964
abc 1964

s p ä r r n i n g
knipning

Versaler (minska 10%)

Han var SACO-medlem men…
Han var SACO-medlem men…

Parknipning (Pair kerning)

Tyfon

Tyfon

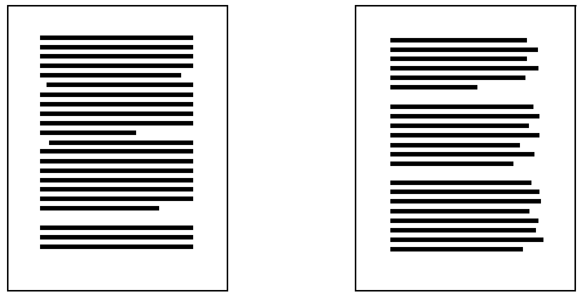
VALTAND
VALTAND

Avfall
Avfall

Automatisk spärrning

Man bör akta sig för ordbehandlare som automatiskt spärrar bokstäver för att undvika avstavningar när man använder högerjusterad marginal. Det ger e t t oprofessionellt intryck och en svårläst text eftersom vuxna människor inte läser enskilda bokstäver, utan ”ordbilder”.

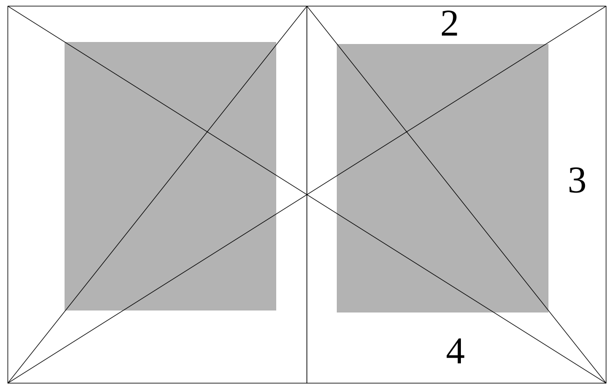
Marginaler



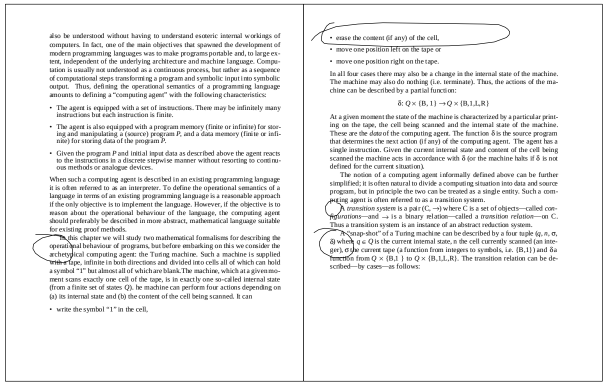
Spara på krutet!!!

Använd typografiska signaler endast när de verkligen behövs

Satsytan



Stycke och sidbrytningar



also be understood without having to understand esoteric internal workings of computers. In fact, one of the main objectives that spawned the development of modern programming languages was to make programs portable and, to large extent, independent of the underlying architecture and machine language. Computation is usually not understood as a continuous process, but rather as a sequence of computational steps transforming a program and symbolic input into symbolic output. Thus, defining the operational semantics of a programming language amounts to defining a "computing agent" with the following characteristics:

- The agent is equipped with a set of instructions. There may be infinitely many instructions but each instruction is finite.
- The agent is also equipped with a program memory (finite or infinite) for storing and manipulating a (source) program *P*, and a data memory (finite or infinite) for storing data of the program *P*.
- Given the program *P* and initial input data as described above the agent reacts to the instructions in a discrete stepwise manner without resorting to continuous methods or analogue devices.

When such a computing agent is described in an existing programming language it is often referred to as an interpreter. To define the operational semantics of a language in terms of an existing programming language is a reasonable approach if the only objective is to implement the language. However, if the objective is to reason about the operational behaviour of the language, the computing agent should preferably be described in more abstract, mathematical language suitable for existing proof methods.

In this chapter we will study two mathematical formalisms for describing the operational behaviour of programs, but before embarking on this we consider the archetypical computing agent: the Turing machine. Such a machine is supplied with a tape, infinite in both directions and divided into cells all of which can hold a symbol "1" but almost all of which are blank.The machine, which at a given moment scans exactly one cell of the tape, is in exactly one so-called internal state (from a finite set of states *Q*). he machine can perform four actions depending on (a) its internal state and (b) the content of the cell being scanned. It can

- write the symbol "1" in the cell,

- erase the content (if any) of the cell,
- move one position left on the tape or
- move one position right on the tape.

In all four cases there may also be a change in the internal state of the machine. The machine may also do nothing (i.e. terminate). Thus, the actions of the machine can be described by a partial function:

$$\delta: Q \times \{B, 1\} \rightarrow Q \times \{B,1,L,R\}$$

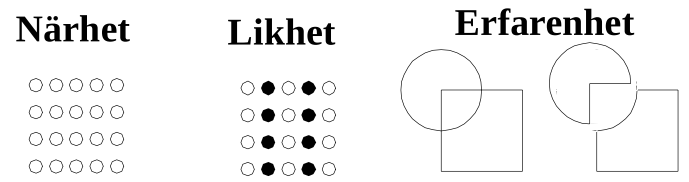
At a given moment the state of the machine is characterized by a particular printing on the tape, the cell being scanned and the internal state of the machine. These are the *data* of the computing agent. The function δ is the source program that determines the next action (if any) of the computing agent. The agent has a single instruction. Given the current internal state and content of the cell being scanned the machine acts in accordance with δ (or the machine halts if δ is not defined for the current situation).

The notion of a computing agent informally defined above can be further simplified; it is often natural to divide a computing situation into data and source program, but in principle the two can be treated as a single entity. Such a computing agent is often referred to as a transition system.

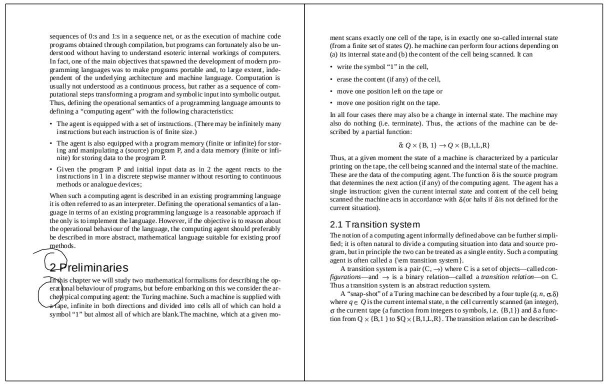
A *transition system* is a pair (C, →) where C is a set of objects—called *configurations*—and → is a binary relation—called a *transition relation*—on C. Thus a transition system is an instance of an abstract reduction system.

A "snap-shot" of a Turing machine can be described by a four tuple (*q*, *n*, σ, δ) where $q \in Q$ is the current internal state, n the cell currently scanned (an integer), σ the current tape (a function from integers to symbols, i.e. {B,1}) and δ a function from $Q \times \{B,1\}$ to $Q \times \{B,1,L,R\}$. The transition relation can be described—by cases—as follows:

Samhörighet



Rubriker



sequences of 0:s and 1:s in a sequence net, or as the execution of machine code programs obtained through compilation, but programs can fortunately also be understood without having to understand esoteric internal workings of computers. In fact, one of the main objectives that spawned the development of modern programming languages was to make programs portable and, to large extent, independent of the underlying architecture and machine language. Computation is usually not understood as a continuous process, but rather as a sequence of computational steps transforming a program and symbolic input into symbolic output. Thus, defining the operational semantics of a programming language amounts to defining a "computing agent" with the following characteristics:

- The agent is equipped with a set of instructions. (There may be infinitely many instructions but each instruction is of finite size.)
- The agent is also equipped with a program memory (finite or infinite) for storing and manipulating a (source) program P, and a data memory (finite or infinite) for storing data to the program P.
- Given the program P and initial input data as in 2 the agent reacts to the instructions in 1 in a discrete stepwise manner without resorting to continuous methods or analogue devices;

When such a computing agent is described in an existing programming language it is often referred to as an interpreter. Defining the operational semantics of a language in terms of an existing programming language is a reasonable approach if the only is to implement the language. However, if the objective is to reason about the operational behaviour of the language, the computing agent should preferably be described in more abstract, mathematical language suitable for existing proof methods.

2 Preliminaries

In this chapter we will study two mathematical formalisms for describing the operational behaviour of programs, but before embarking on this we consider the archetypical computing agent: the Turing machine. Such a machine is supplied with a tape, infinite in both directions and divided into cells all of which can hold a symbol "1" but almost all of which are blank.The machine, which at a given moment scans exactly one cell of the tape, is in exactly one so-called internal state (from a finite set of states *Q*). he machine can perform four actions depending on (a) its internal state and (b) the content of the cell being scanned. It can

- write the symbol "1" in the cell,
- erase the content (if any) of the cell,
- move one position left on the tape or
- move one position right on the tape.

In all four cases there may also be a change in internal state. The machine may also do nothing (i.e. terminate). Thus, the actions of the machine can be described by a partial function:

$$\delta: Q \times \{B, 1\} \rightarrow Q \times \{B,1,L,R\}$$

Thus, at a given moment the state of a machine is characterized by a particular printing on the tape, the cell being scanned and the internal state of the machine. These are the data of the computing agent. The function δ is the source program that determines the next action (if any) of the computing agent. The agent has a single instruction: given the current internal state and content of the cell being scanned the machine acts in accordance with δ(or halts if δis not defined for the current situation).

2.1 Transition system

The notion of a computing agent informally defined above can be further simplified; it is often natural to divide a computing situation into data and source program, but in principle the two can be treated as a single entity. Such a computing agent is often called a {\em transition system}.

A transition system is a pair (C, →) where C is a set of objects—called *configurations*—and → is a binary relation—called a *transition relation*—on C. Thus a transition system is an abstract reduction system.

A "snap-shot" of a Turing machine can be described by a four tuple (*q*, *n*, σ,δ) where $q \in Q$ is the current internal state, n the cell currently scanned (an integer), σ the current tape (a function from integers to symbols, i.e. {B,1}) and δ a function from Q × {B,1 } to $Q × {B,1,L,R}. The transition relation can be described-

Rubriker

- Motsvarar rubriknivån avsnittets betydelse?
- Ger rubrikerna en sammanfattning av texten?
- Ingen punkt efter rubrik!
- Undvik avstavningar (stryk onödiga ord)!
- Högst tre rubriknivåer (inklusive kapitel)!
- Undvik underrubrik direkt efter rubrik!
- Inga (under)rubriker om det bara finns ett (under)avsnitt!

Tänk på läsbarheten

- Teckensnitt. Välj så få som möjligt. (Gärna bara två.)
- Teckenstorlek (10–12 punkter).
- Radavstånd. Gärna två punkter större än teckenstorlek.
- Spaltbredd. Absolut inte mer än 70 tecken per rad!
- Ransonera typografiska markeringar.
- Dela upp texten med rubriker och styckeindelning.

Numrering

Numrera avsnitt, kapitel etc.
om de refereras.

Numrera kapitelvis.

Numrera definitioner, satser etc.
i en löpande serie.

Listor

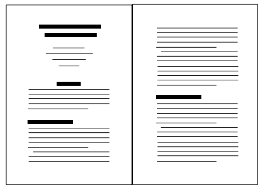
Alla listelement på samma form!

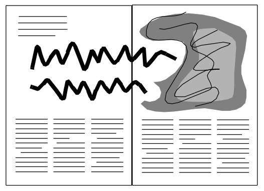
Använd diskreta ”bomber”!

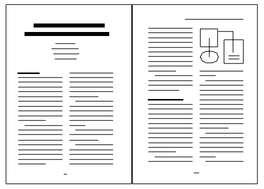
Extra utrymme runt varje
element i listan

Använd ”och” och ”eller” om
oklarhet kan uppkomma

Sidlayout







Bindestreck och tankstreck

Semi-structured (s.k. divis)

see pages 5–8

Använd *tankstreck* – inte *divis*.

The English version—which is
twice as long—is called *em-dash*.

Kolon

Sedan skrek hon: ”Kom hit!”

Flaggan hade två färger: blå och gul.

S:t Anna

6:e juni

49:50 kr

LO:s ordförande

Semikolon

Han slocknade totalt efter trehundra meter;
det var som om luften gick ur honom.

She saw:
- paintings by Renoir, Monet and Gauguin;
- statues by Rodin and Picasso.

Hon såg tavlor av Renoir, Monet och
Gauguin; statyer av Rodin och Picasso.

Noter

Huvudregel
Texten ska kunna läsas utan onödiga avbrott

Placera inte nödvändig information
i slutnoter eller fotnoter!

Referenser

In logic programming, tabling [18, 2] is emerging as a powerful evaluation technique. Tabling systems evaluate programs by re-

In logic programming, tabling (see Brown [2] and Green et al. [18]) is emerging as a powerful evaluation technique. Tabling

***

[20] U. Nilsson, Abstract Interpretation: A Kind of Magic. *Theoretical Computer Science*, 56(2), 234–256, 1995.

Radavståndet (kägel)

Ett för litet radavstånd ger ett oprofessionellt intryck och svårläst text. Man bör se till att radavståndet är större än ordmellanrummen.

Ett för litet radavstånd ger ett oprofessionellt intryck och svårläst text. Man bör se till att radavståndet är större än ordmellanrummen.

Proportioner

Huvudrubrik (24 p)

Underrubrik (16 p.)

Brödtext (10 p.)

Inga siffror eller formler först

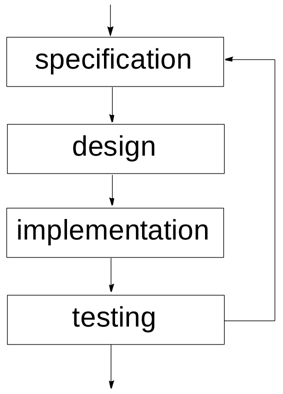
Skriv inte:

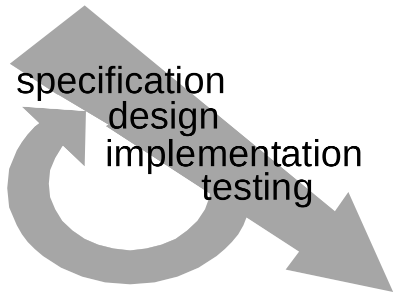
> π är ett irrationellt tal. 3.14 används ofta som approximation för π.

Skriv istället:

> Talet π är irrationellt. Ofta används 3.14 som approximation för π.

Bilder (forts.)





Bilder

Förklara alltid bilden i brödtexten!

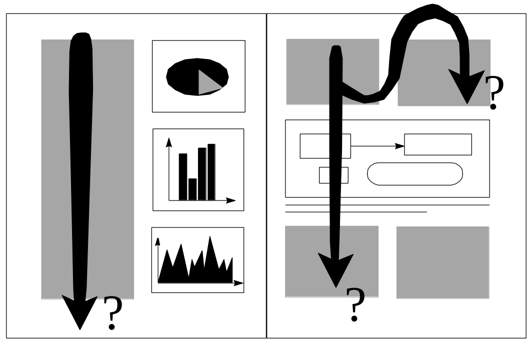
Uppge eventuell källa!

Skriv alltid en kortfattad bildtext!

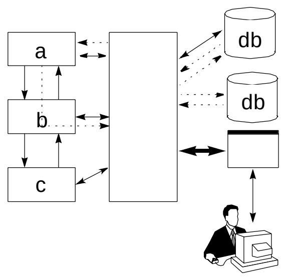
Tryck inte in för mycket i en enda bild!

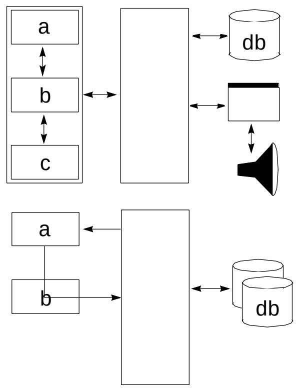
Undvik ”clip-art”!

Textflödet

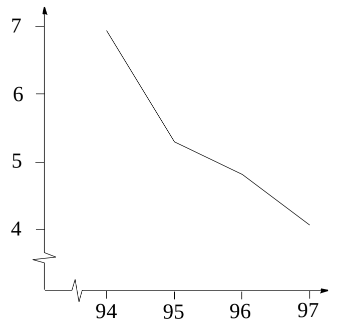


Bilder





Tusen ord?


7
6
5
4
94
95
96
97