# Exploring Life through Logic Programming

Alessandro Dal Palú  Agostino Dovier
Andrea Formisano  Enrico Pontelli

Dept. Computer Science, New Mexico State University, USA

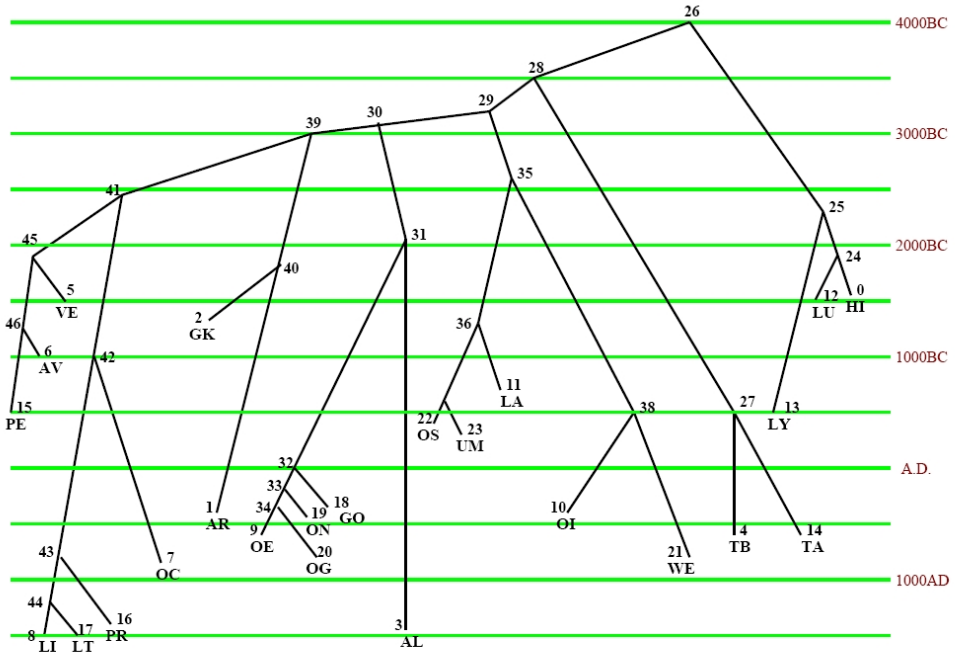
University of Udine, Italy
December 2015

# Phylogenetics

# Phylogenetic trees

## Basics

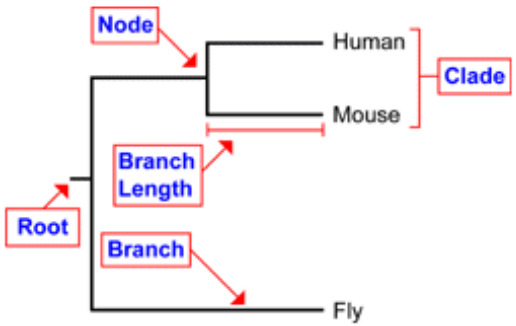
- A phylogeny describes evolutionary relationships among entities.
- Comparative biology: investigates similarities and differences
- More reliable than pattern matching
  - Provide insights into relationships among organisms (*species trees*)
  - Provide insights into history and evolution of genes (*gene trees*)
- Applied outside biology: e.g.,
  - Epidemiology (development of pandemics, patterns of disease transmission)
  - Ecology (population structure, population at risk)
  - Indo-European languages
  - Forensics (e.g., detect smuggled animals)



# Phylogenetic trees

## Basics

- A phylogeny is a "tree", which estimates the "historical" connections between species or genes that they carry.
- Parts of a phylogenetic tree include:
  - The "tips" of the tree are the *taxa* in the study.
  - Taxa may be at any taxonomic level - orders, species, populations, etc.
  - These taxa may be called OTUs, or Operational Taxonomic Units.
  - The lines within the tree are called the *branches.*
  - Some trees will have a basal node
  - A grouping of an ancestor and all of its descendants is known as a *clade.*
- The "goal" of phylogenetic analysis is to recover "bifurcating" trees



# Phylogenetic trees

## Basics of Model

- Given a set $L$ of elementary taxonomic units,
  - $L = \{English, German, French, Spanish, Italian\}$
  - $L = \{dog, cat, horse, chicken\}$
- A set $C$ of characters is assigned to each element of $L$ (e.g., characters "hand" and "father", or characters "number of legs", "length of the tail", etc.)
- Characters can be mapped to valued from a finite set
  - Character hand has values $1 =$ *hand* and $2 =$ *mano/main*
  - Character father has values $1 =$ *father/padre* and $2 =$ *vater/père*
- Each element in $L$ is assigned a value for each character.
- Let us focus on Boolean characters

# Phylogenetic tree reconstruction

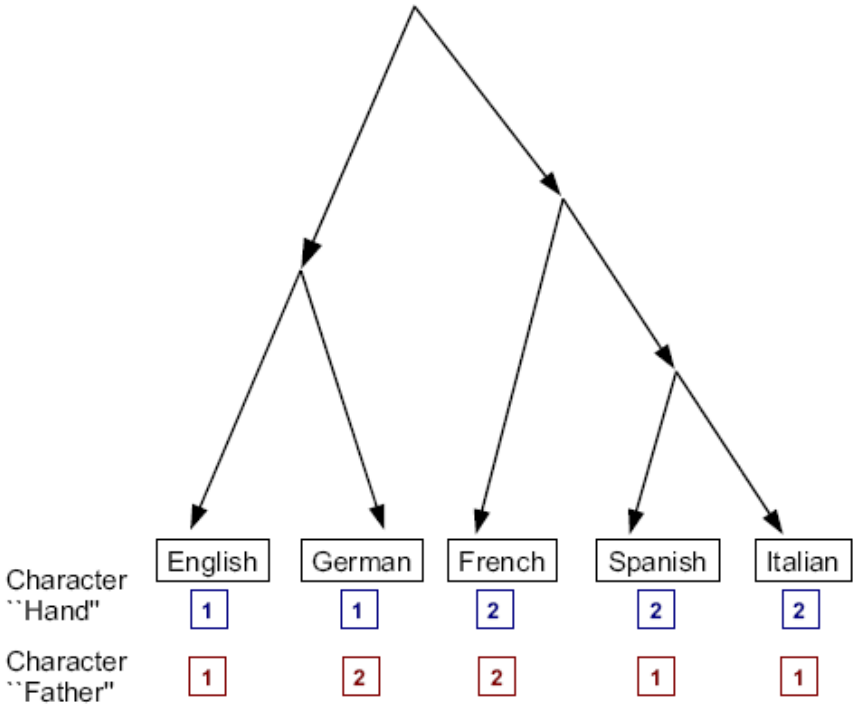
- A phylogeny

$$(V, E, L, C, D, f)$$

for a set $L$ of taxa is a
  - Finite binary tree $(V, E)$ with leaves $L \subseteq V$
  - Two finite sets $C = \{c_1, \ldots, c_k\}$ and $D = \{D_1, \ldots, D_k\}$ and a function $f : L \times C \longrightarrow D$.
- $V \setminus L$ describes the *ancestral units* and $E$ the evolutionary relationships.
- $C$ is the set of characters, and $D$ contains their domain values (also knows as states)
- $f$ labels every leaf $v \in L$ by assigning a state $f(v, i)$ for each character $i \in C$

# Phylogenetic trees

## Example (from Erdem11)

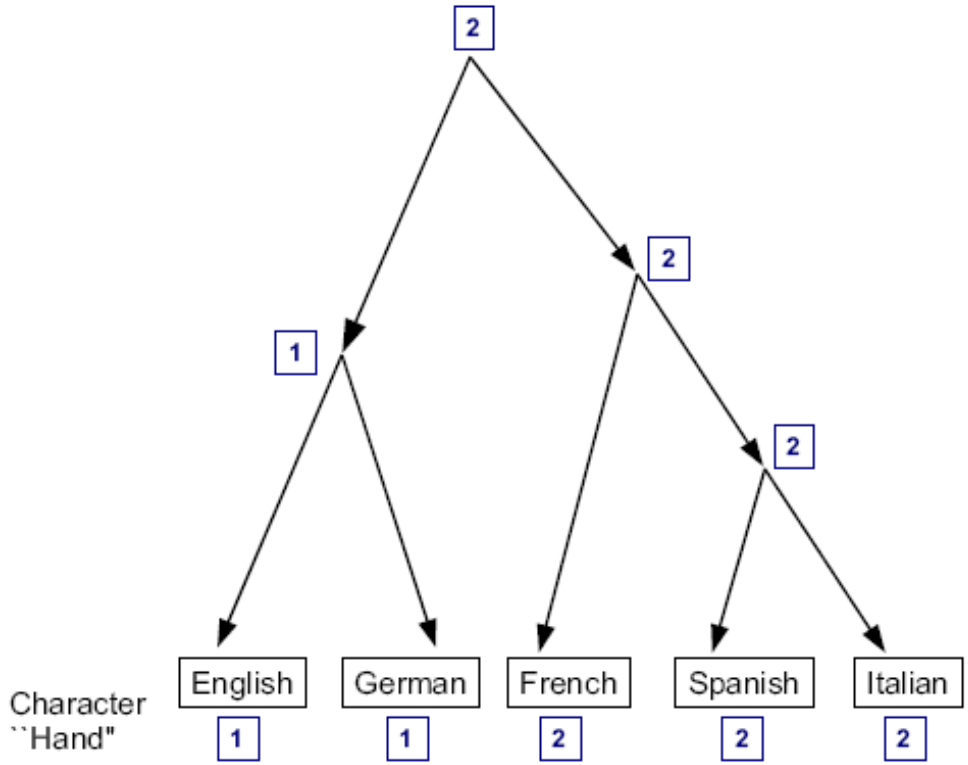



A phylogeny $(V, E, L, C, D, f)$ where $L = \{\text{English}, \text{German}, \text{French}, \text{Spanish}, \text{Italian}\}$ (taxa) $C = \{\text{Hand}, \text{Father}\}$ (characters), $D_{\text{Hand}} = D_{\text{Father}} = \{1, 2\}$ (states).

# Phylogenetic trees

## Example (from Erdem 2011)

- A character $i \in C$ is compatible with a phylogeny if the taxa that present the same value for $i$ are connected by a subtree.

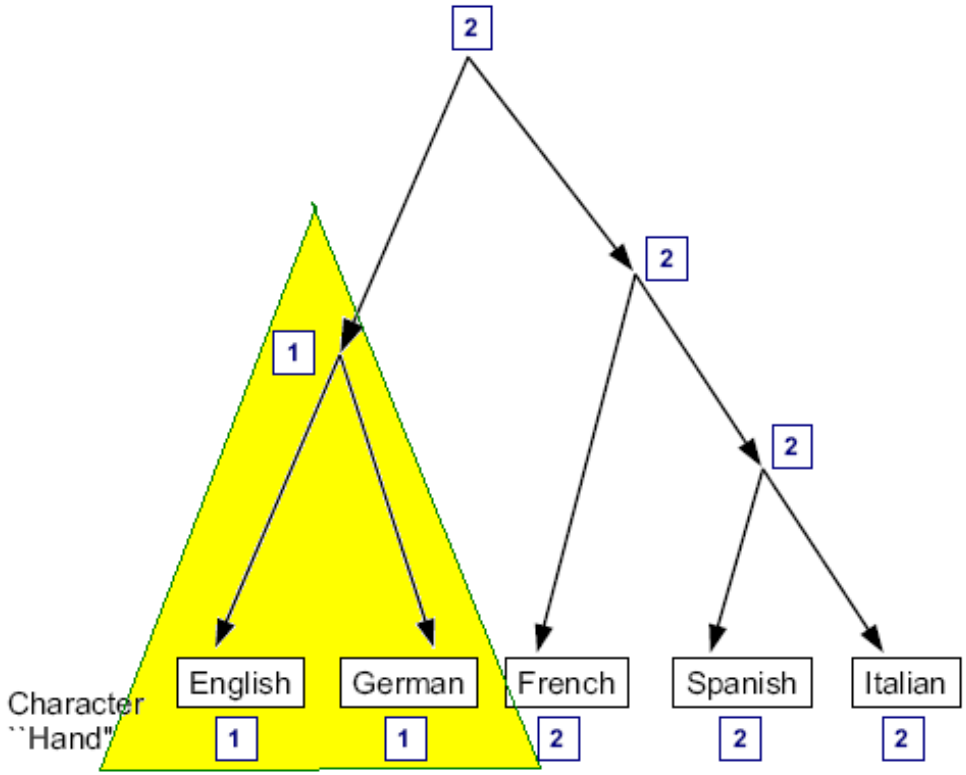



Character Hand is compatible with the above tree

# Phylogenetic trees

## Example (from Erdem 2011)

- A character $i \in C$ is compatible with a phylogeny if the taxa that present the same value for $i$ are connected by a subtree.

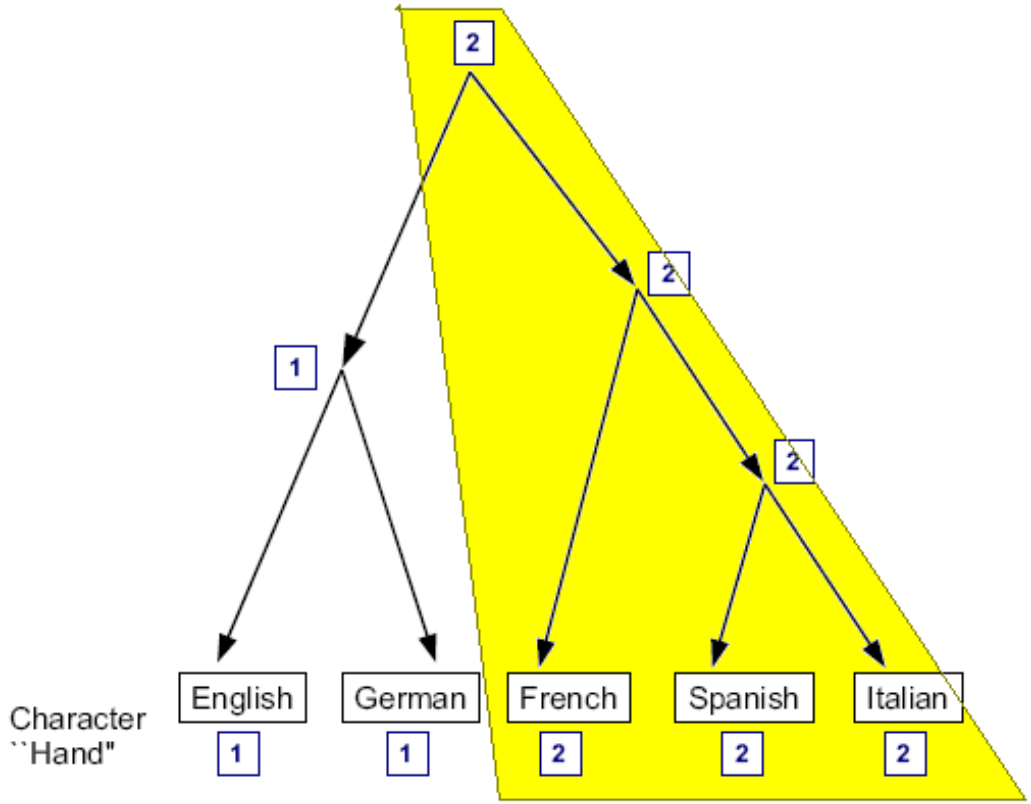



Character Hand is compatible with the above tree

# Phylogenetic trees

## Example (from Erdem 2011)

- A character $i \in C$ is compatible with a phylogeny if the taxa that present the same value for $i$ are connected by a subtree.

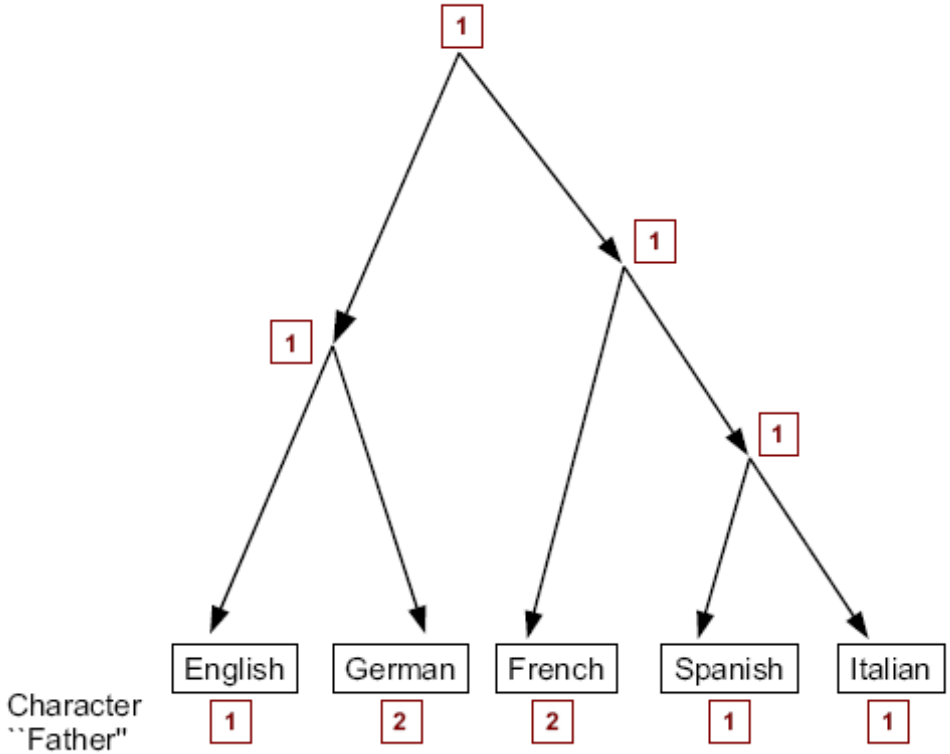



Character Hand is compatible with the above tree

# Phylogenetic trees

## Example (from Erdem 2011)

- A character $i \in C$ is compatible with a phylogeny if the taxa that present the same value for $i$ are connected by a subtree.
- Otherwise it is incompatible

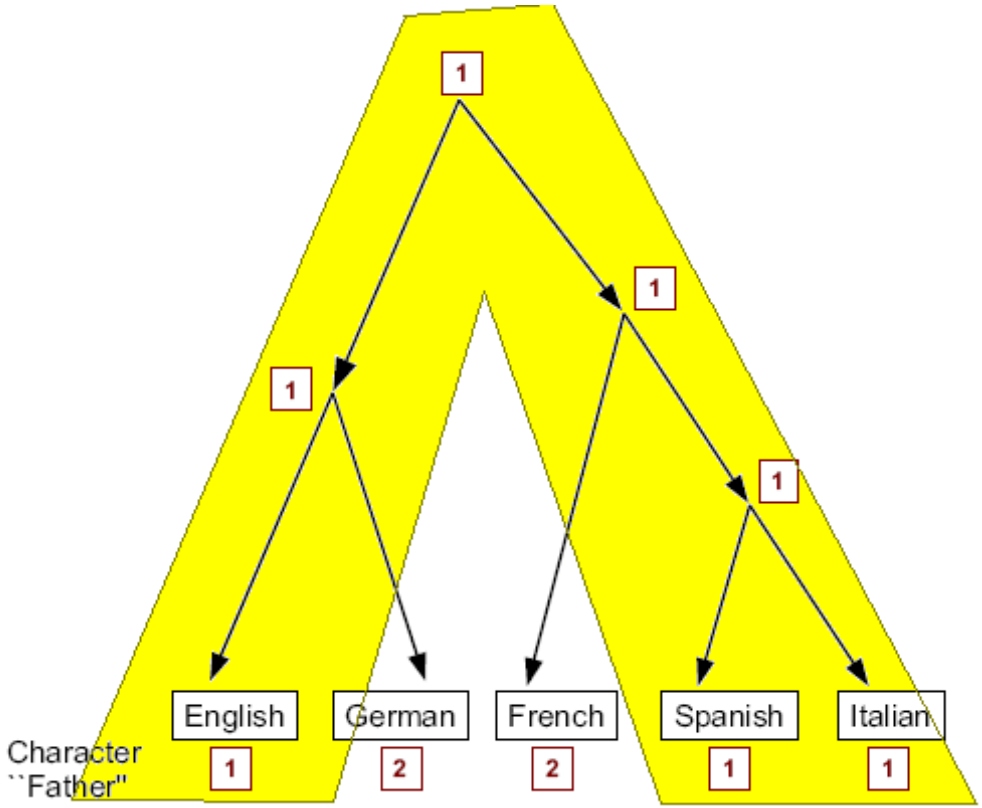



Character Father is incompatible with the above tree

# Phylogenetic trees

## Example (from Erdem 2011)

- A character $i \in C$ is compatible with a phylogeny if the taxa that present the same value for $i$ are connected by a subtree.
- Otherwise it is incompatible

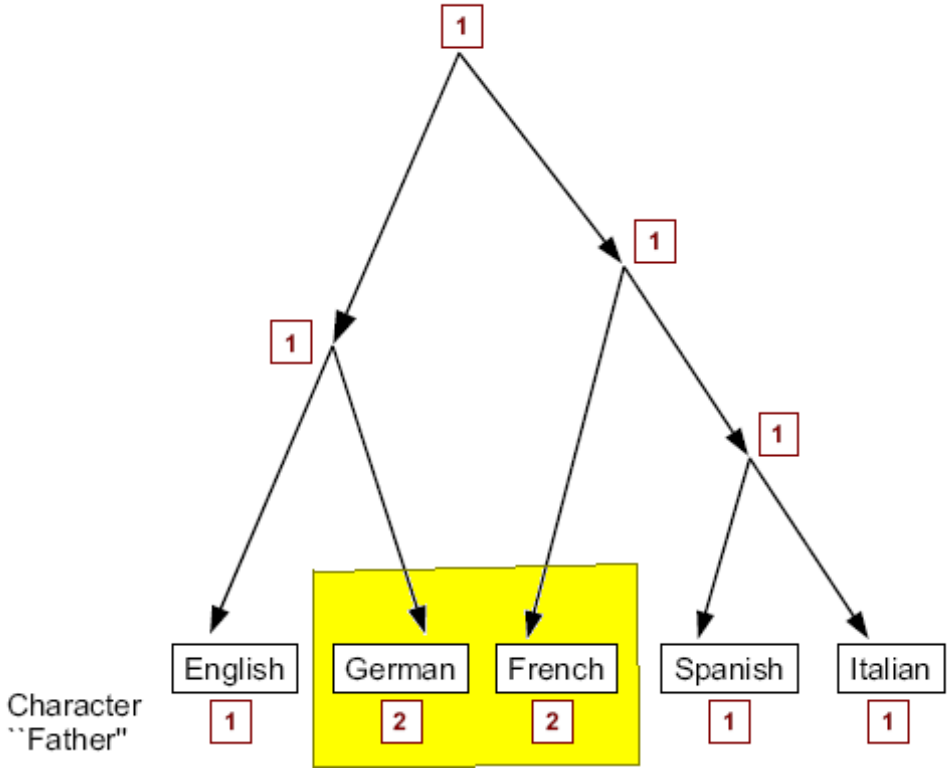



Character Father is incompatible with the above tree

# Phylogenetic trees

## Example (from Erdem 2011)

- A character $i \in C$ is compatible with a phylogeny if the taxa that present the same value for $i$ are connected by a subtree.
- Otherwise it is incompatible




Character Father is incompatible with the above tree

# Phylogenetic trees

## $k$-incompatibility

- The above subtree requirement implicitly states that when a character changes (in the evolution) it never go back to the previous value (Camin-Sokal). Moreover, that the change occurs in a unique place (Dollo).

# Phylogenetic trees

## $k$-incompatibility

- The above subtree requirement implicitly states that when a character changes (in the evolution) it never go back to the previous value (Camin-Sokal). Moreover, that the change occurs in a unique place (Dollo).

### $k$-INCOMPATIBILITY PROBLEM

Given sets $L$ (taxa/leaves), $C$ (characters), and $D$ (states), a function $f : L \times C \longrightarrow D$, and $k \in \mathbb{N}$, decide the existence of a phylogeny $(V, E, L, C, D, f)$ with at most $k$ incompatible characters.

- This problem is NP-complete (Day, Sankoff 1986).
- The number of possible phylogenies is exponential in $L$

# ASP Encoding

## Facts

- Represent taxa by numbers:

```
taxon(1..N).
```

- Represent characters $C = \{c_1, \ldots, c_k\}$:

```
character(c_i).
```

- Represent extensionally the domain $D_i$ of character $c_i$, for each $v \in D_i$:

```
domain(c_i,v).
```

- Represent extensionally the function $f$: for each $\ell \in L$, $c \in C$, such that $f(\ell, c) = v$:

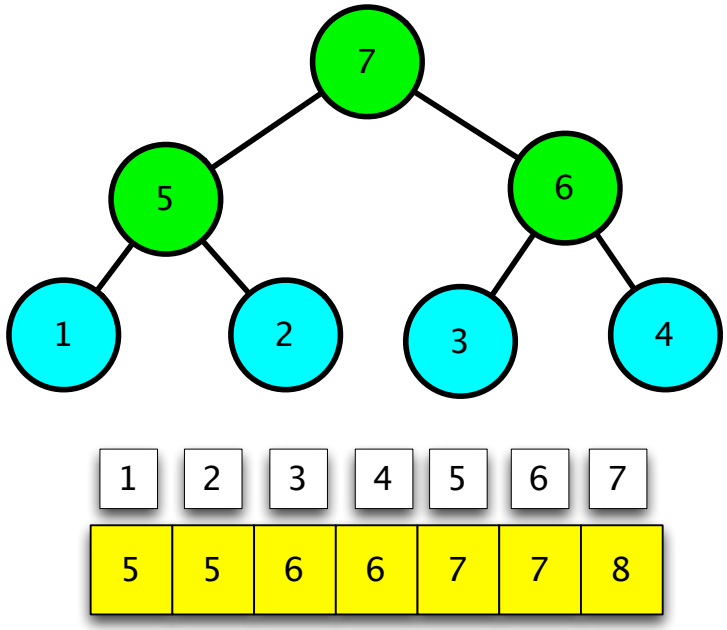
```
f(i,c,v).
```

# ASP Encoding

## Binary tree construction

```
(1)  num_of_taxa(N) :- N = #count{ L : taxon(L) }.
(2)  node(1..2*N-1) :- num_of_taxa(N).
```

- Tree nodes: $1 \ldots 2N-1$
- Taxa: $1 \ldots N$
- Internal (ancestor) nodes: $N+1 \ldots 2N-1$;
- Root is $2N-1$



# ASP Encoding

## Binary tree construction

```
(1)  num_of_taxa(N) :- N = #count{ L : taxon(L) }.
(2)  node(1..2*N-1) :- num_of_taxa(N).

(3)  2 { edge(I,J): node(J)} 2 :-
                node(I), not taxon(I).
```

- Each internal node has exactly two children
- Internal node is a node, but not a taxon

# ASP Encoding

## Binary tree construction

```
(1)  num_of_taxa(N) :-  N =  #count{ L : taxon(L) }.
(2)  node(1..2*N-1)  :-  num_of_taxa(N).

(3)  2 { edge(I,J): node(J)} 2 :-
                node(I), not taxon(I).

(4)  :- node(I), node(J), edge(I,J), I < J.
```

- Children are of smaller index
- Constraint: edges can't link a greater index

# ASP Encoding

## Binary tree construction

```
(1)  num_of_taxa(N) :- N = #count{ L : taxon(L) }.
(2)  node(1..2*N-1) :- num_of_taxa(N).

(3)  2 { edge(I,J): node(J)} 2 :-
                node(I), not taxon(I).

(4)  :- node(I), node(J), edge(I,J), I < J.

(5)  1 { edge(I,J): node(I) } 1 :-
              node(J), num_of_taxa(N), J < 2*N-1.
```

- Each node but the root (=2*N-1) has 1 father

# ASP Encoding

## Compatibility computation

```
(6)   1 { h(I,C,V): domain(C,V) } 1   :-
                node(I), character(C).
(7)   h(I,C,V)   :- taxon(I), f(I,C,V).
```

- extend f to h, in order to cover inner nodes
- h is a total function (for each node, char there is one assignment V)

# ASP Encoding

## Compatibility computation

```
(6)  1 { h(I,C,V): domain(C,V) } 1  :-
                node(I), character(C).
(7)  h(I,C,V)  :- taxon(I), f(I,C,V).

(8)  char_edge(C,V,A,B) :- node(A), node(B), domain(C,V),
          edge(A,B), h(A,C,V), h(B,C,V).
```

- For each character and original edge, build char_edge.
- They are labeled edges that form new trees on nodes.
- The goal is to determine if we have trees or forests.
- A forest shows that incompatible values for char are used.

# ASP Encoding

## Compatibility computation

```
(9)  no_root(C,V,A) :- node(A), node(B), domain(C,V),
                       h(A,C,V), char_edge(C,V,B,A).
(10) root(C,V,A) :- node(A), domain(C,V),
                    h(A,C,V), not no_root(C,V,A).
(11) two_roots(C,V) :- node(A), node(B), A < B,
                       domain(C,V),
                       root(C,V,A), root(C,V,B).
```

- For each character and value, if there is an incoming char_edge on A, A is no_root for that tree.
- For each character and value, the node A is a root is there are no no_root
- The goal is to determine if we have trees or forests.
- If there are two roots, we have a forest for char C and value V.

# ASP Encoding

## Compatibility computation

```
(12) partially_compatible(C,V) :-
                domain(C,V), not two_roots(C,V).
(13) {compatible(C)} :- character(C).
(14) :- character(C),  compatible(C), domain(C,V),
                not partially_compatible(C,V).
```

- For each character and value, we have (partial) compatibility if there is a tree, i.e., there is no two_roots(_,_).
- We define a number of compatible characters (unbound cardinality)
- Constraint: C can't be compatible and not partially_compatible for one value

# ASP Encoding

## Compatibility computation

```
(15) compats(N) :- N = #count{ C:compatible(C) }.
(16) #maximize { N:compats(N) }.
```

- Counts the number of compatible characters
- Maximize the count

# ASP Encoding

## Compatibility computation

```
(15) compats(N) :- N = #count{ C:compatible(C) }.
(16) #maximize { N:compats(N) }.
```

- Counts the number of compatible characters
- Maximize the count

For the decisional version (at most $k$ compatible characters)

```
(15-16) k { compatible(C) : character(C) }.
```

# (Some) References


- J. Felsenstein, Inferring Phylogenies. Sinauer Inc., 2004.
- Day, W.H.E., Johnson D.S., Sankoff, D. The Computational complexity of Inferring Rooted Phylogenies by Parsimony. Math. Biosciences 81:33–42, 1986.
- Day, W.H.E., Sankoff, D. Computational complexity of Inferring Phylogenies by Compatibility. *Systematic Zoology* 35(2):224–229, 1986.
- Erdem E., Lifschitz V., Nakhleh L., Ringe D. Reconstructing the Evolutionary History of Indo-European Languages Using Answer Set Programming. PADL 2003: 160-176.
- Thomas Schiex et al. Papers on *complex pedigree reconstructions* using weighted constraint satisfaction. In WCB 05, WCB 06, WCB 07.
- Moore N.C.A., and Prosser P. The Ultrametric Constraint and its Application to Phylogenetics. *J. Artif. Intell. Res.* 32:901–938, 2008 (also in WCB 06)
- Erdem E. Applications of Answer Set Programming in Phylogenetic Systematics MG65, LNCS 6565, 2011.