

Compiler construction in4303 – lecture 6



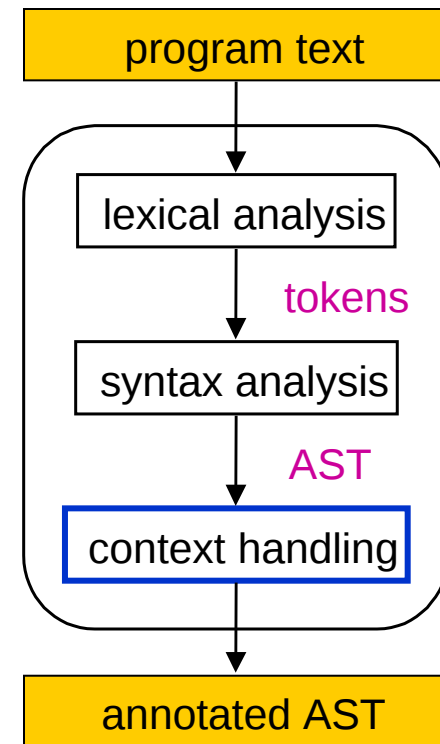
AST processing
attribute grammars

Chapter 3.1

Overview

- context handling
- annotating the AST
 - attribute grammars
 - manual methods

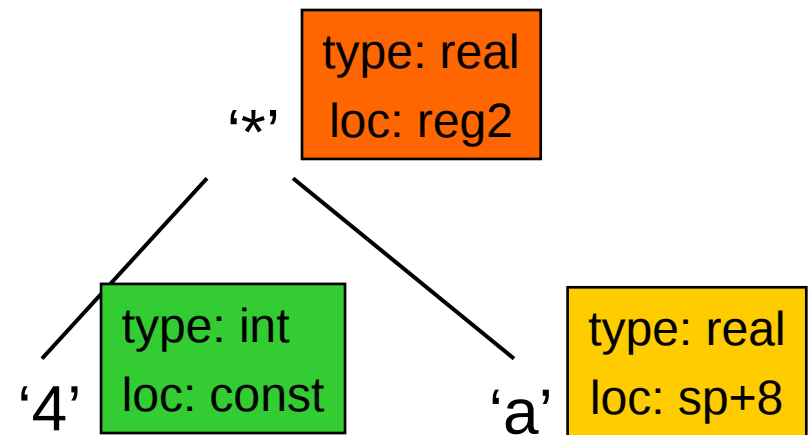
```
expr : expr '+' expr    { $$ = $1 + $3; }  
      | expr '*' expr    { $$ = $1 * $3; }  
      | '(' expr ')'     { $$ = $2; }  
      | DIGIT  
      ;
```



Attribute grammars

- **formal attributes** are associated with each grammar symbol

- type
- location
- context



- **inherited** attributes
- **synthesized** attributes

Attribute grammars

- **attribute evaluation** rules are associated with each production

Constant_definition(**INH** old symbol table, **SYN** new symbol table) →
'CONST' Defined_identifier '=' Expression ';'

ATTRIBUTE RULES:

SET Expression . symbol table **TO**

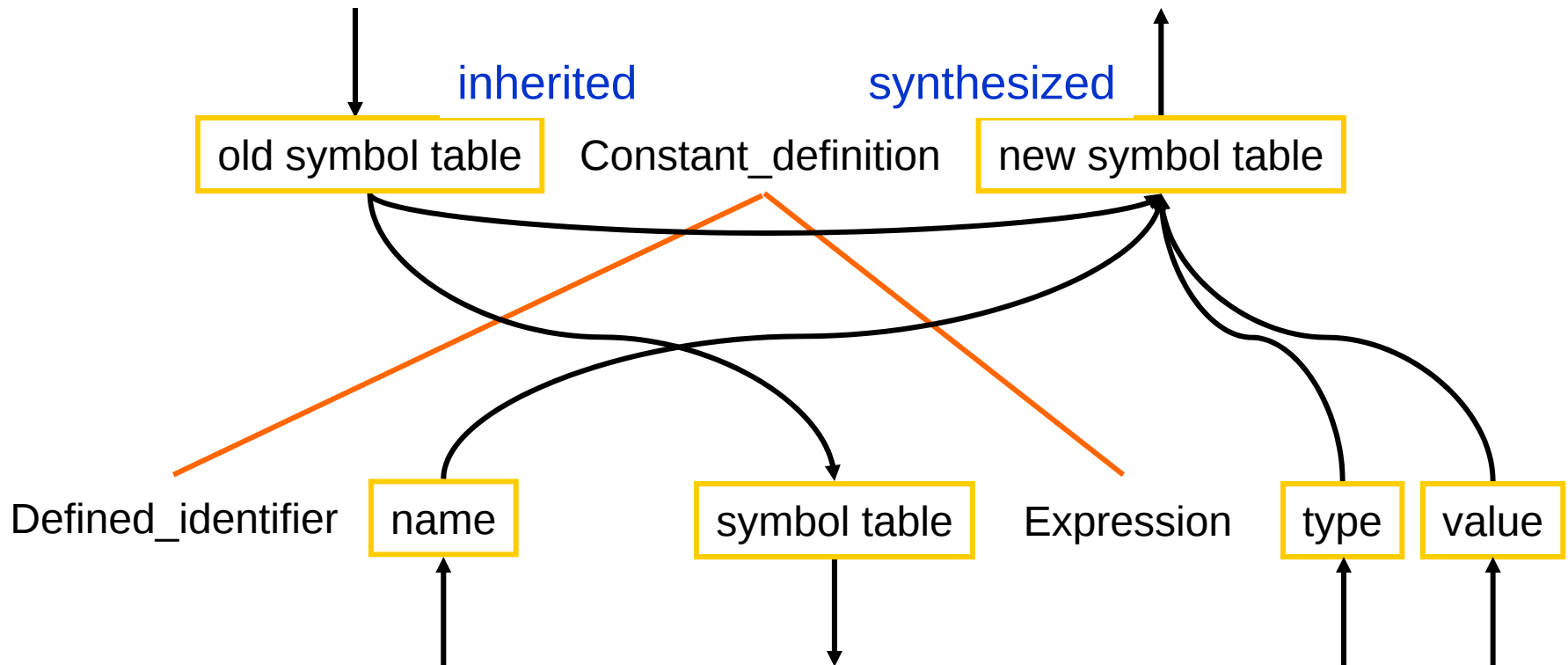
Constant_definition. old symbol table;

SET Constant_definition. new symbol table **TO**

Updated symbol table(Constant_definition. old symbol table,
Defined_identifier. name, Expression. type, Expression. value);

Dependency graph

Constant_definition(INH old symbol table, SYN new symbol table) →
'CONST' Defined_identifier '=' Expression ';'



Concise evaluation rules

Constant_definition(INH old symbol table, SYN new symbol table) →

'CONST' Defined_identifier '=' Expression ';'

ATTRIBUTE RULES:

SET Expression . symbol table TO

Constant_definition. old symbol table;

SET Constant_definition. new symbol table TO

Updated symbol table(Constant_definition. old symbol table,
Defined_identifier. name, Expression. type, Expression. value);

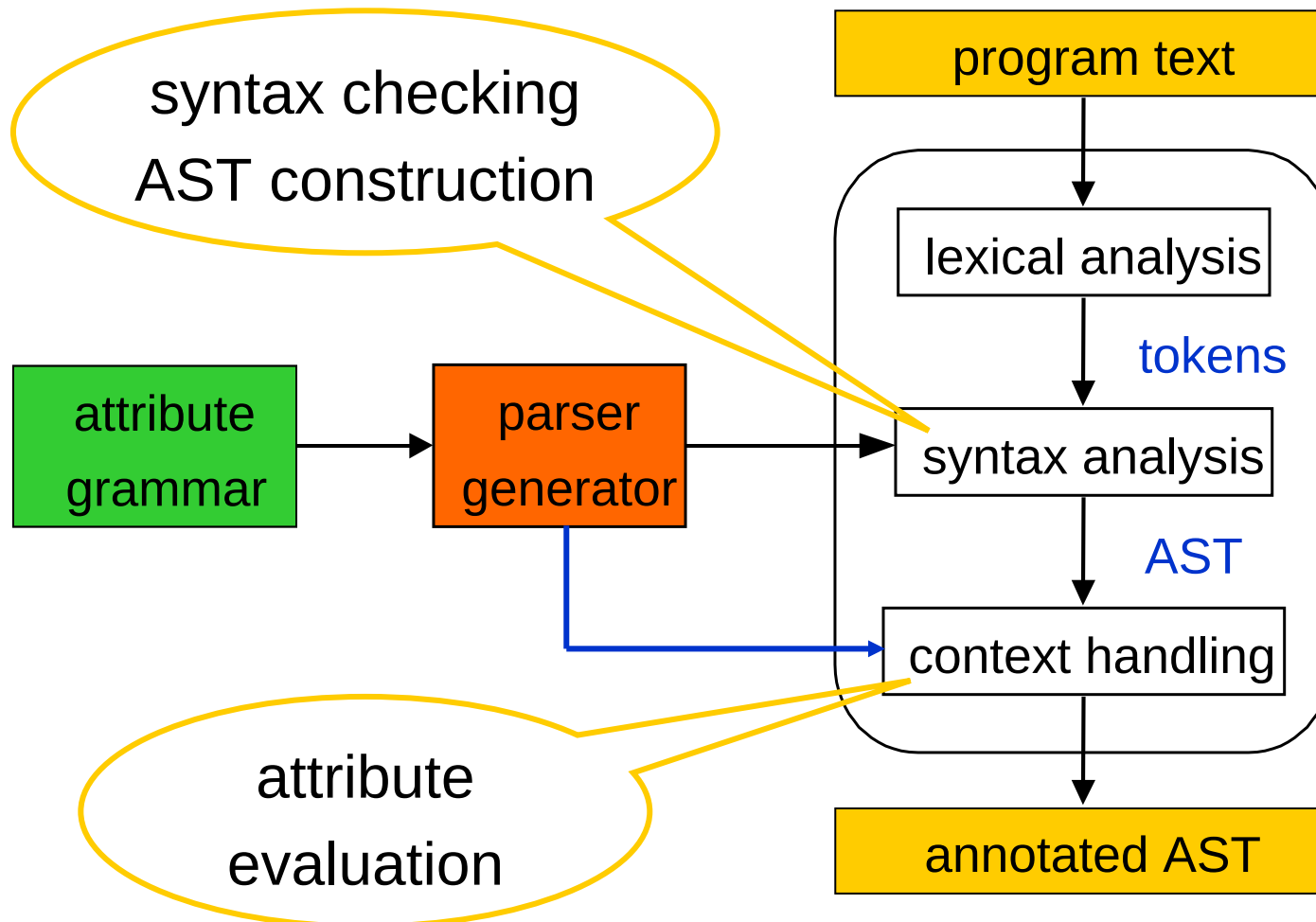
Constant_definition(INH old symtab, SYN new symtab) →

'CONST' Defined_identifier(name) '=' Expression(old symtab, type, value) ';'

ATTRIBUTE RULES:

SET new symtab TO Updated symbol table(old symtab, name, type, value);

Extended parser generator



Running example

integral numbers in decimal or octal notation

- 11D
- 234O
- 56O
- 789D

Number $\rightarrow$ Digit_Seq Base_Tag

Digit_Seq $\rightarrow$ Digit_Seq Digit | Digit

Digit $\rightarrow$ DIGIT // token, '0'-'9'

Base_Tag $\rightarrow$ 'O' | 'D'

Attribute grammar for integral numbers

Number(SYN value) $\rightarrow$ Digit_Seq(base, value) Base_Tag(base)

Digit_Seq(INH base, SYN value) $\rightarrow$ Digit_Seq(base, value) Digit(base, value)

ATTRIBUTE RULES

SET value TO Digit_Seq.value * base + Digit.value;

Digit_Seq(INH base, SYN value) $\rightarrow$ Digit(base, value)

Digit(INH base, SYN value) $\rightarrow$ DIGIT // token, '0'-'9'

ATTRIBUTE RULES

SET value TO Checked digit value(base, DIGIT.repr[0] – '0');

Base_Tag(SYN base) $\rightarrow$ 'O'

ATTRIBUTE RULES

SET base TO 8;

Base_Tag(SYN base) $\rightarrow$ 'D'

ATTRIBUTE RULES

SET base TO 10;

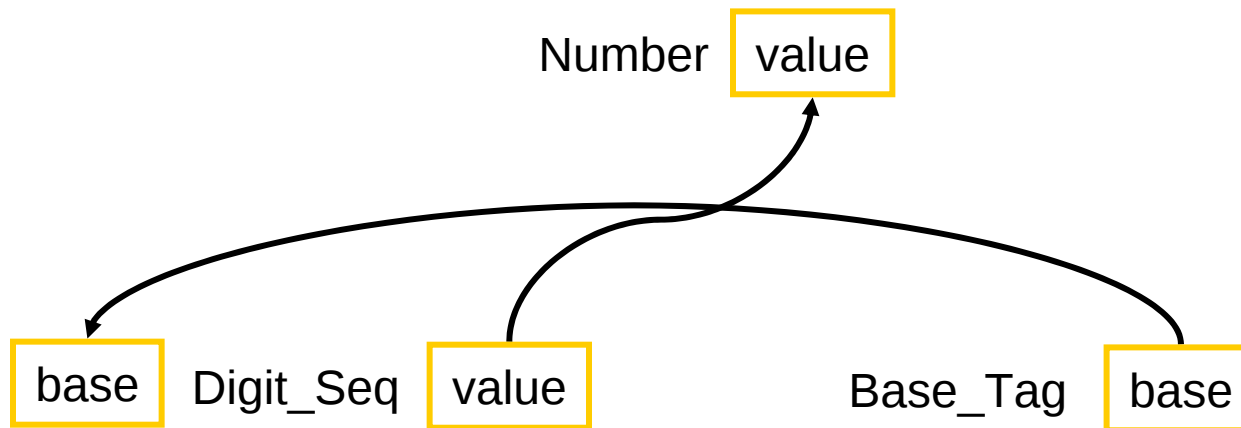
Dependency graphs for integral numbers

Number(SYN value) → Digit_Seq(base, value) Base_Tag(base)

ATTRIBUTE RULES

SET Number.value TO Digit_Seq.value;

SET Digit_Seq.base TO Base_Tag.base;



Exercise (5 min.)



- draw the other dependency graphs for the integral-number attribute grammar

Answers

A thick, horizontal yellow brushstroke underline that spans most of the width of the slide, positioned directly beneath the title.

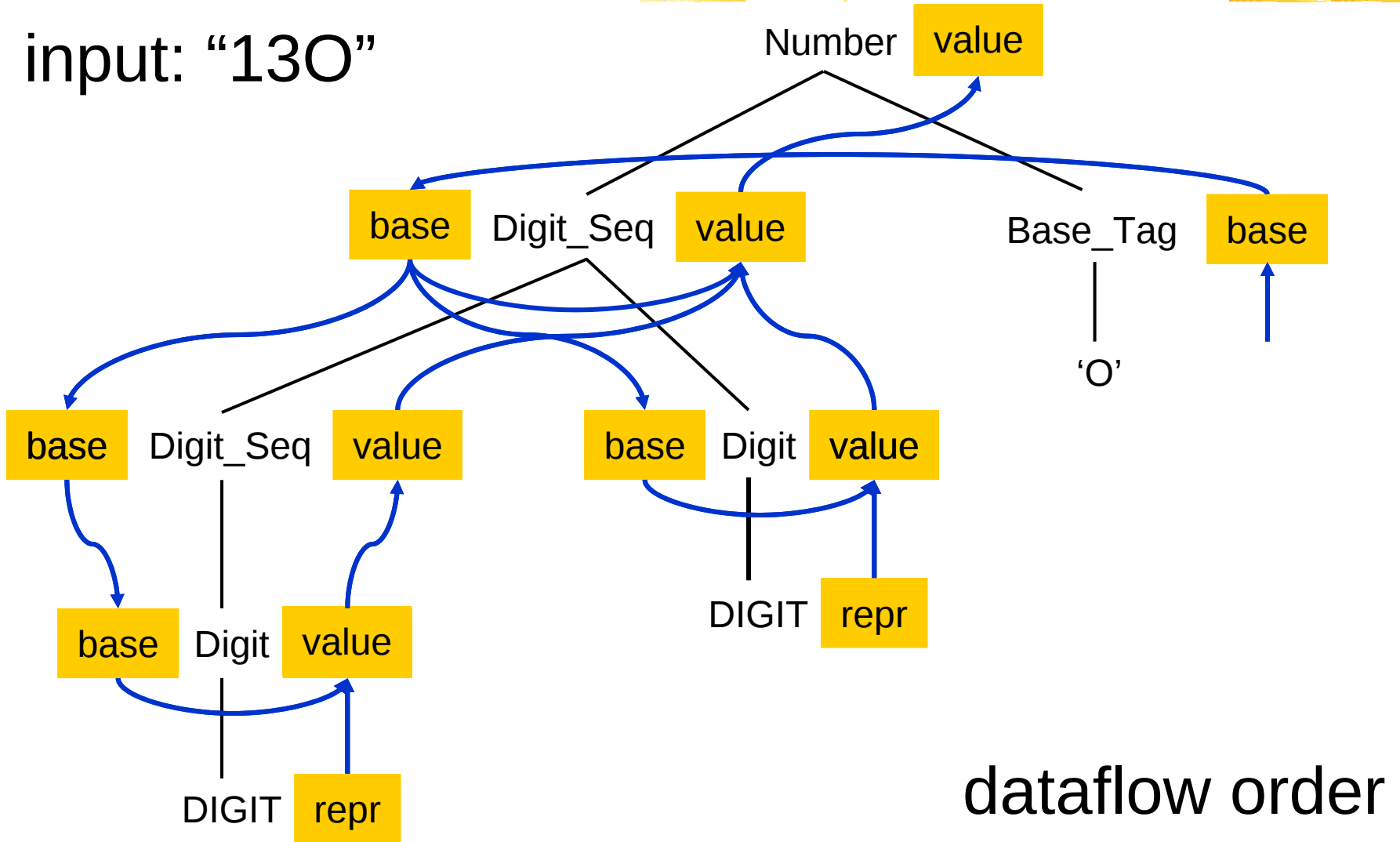
Attribute evaluation



- allocate space for attributes in the nodes of the AST
- fill the attributes of the terminals in the AST (leaf nodes)
- execute the evaluation rules to assign values to attributes (interior nodes)
 - a rule may fire when all input attributes are defined
 - loop until no new values can be assigned

1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023, 2024, 2025, 2026, 2027, 2028, 2029, 2030, 2031, 2032, 2033, 2034, 2035, 2036, 2037, 2038, 2039, 2040, 2041, 2042, 2043, 2044, 2045, 2046, 2047, 2048, 2049, 2050, 2051, 2052, 2053, 2054, 2055, 2056, 2057, 2058, 2059, 2060, 2061, 2062, 2063, 2064, 2065, 2066, 2067, 2068, 2069, 2070, 2071, 2072, 2073, 2074, 2075, 2076, 2077, 2078, 2079, 2080, 2081, 2082, 2083, 2084, 2085, 2086, 2087, 2088, 2089, 2090, 2091, 2092, 2093, 2094, 2095, 2096, 2097, 2098, 2099, 2100, 2101, 2102, 2103, 2104, 2105, 2106, 2107, 2108, 2109, 2110, 2111, 2112, 2113, 2114, 2115, 2116, 2117, 2118, 2119, 2120, 2121, 2122, 2123, 2124, 2125, 2126, 2127, 2128, 2129, 2130, 2131, 2132, 2133, 2134, 2135, 2136, 2137, 2138, 2139, 2140, 2141, 2142, 2143, 2144, 2145, 2146, 2147, 2148, 2149, 2150, 2151, 2152, 2153, 2154, 2155, 2156, 2157, 2158, 2159, 2160, 2161, 2162, 2163, 2164, 2165, 2166, 2167, 2168, 2169, 2170, 2171, 2172, 2173, 2174, 2175, 2176, 2177, 2178, 2179, 2180, 2181, 2182, 2183, 2184, 2185, 2186, 2187, 2188, 2189, 2190, 2191, 2192, 2193, 2194, 2195, 2196, 2197, 2198, 2199, 2200, 2201, 2202, 2203, 2204, 2205, 2206, 2207, 2208, 2209, 2210, 2211, 2212, 2213, 2214, 2215, 2216, 2217, 2218, 2219, 2220, 2221, 2222, 2223, 2224, 2225, 2226, 2227, 2228, 2229, 2230, 2231, 2232, 2233, 2234, 2235, 2236, 2237, 2238, 2239, 2240, 2241, 2242, 2243, 2244, 2245, 2246, 2247, 2248, 2249, 2250, 2251, 2252, 2253, 2254, 2255, 2256, 2257, 2258, 2259, 2260, 2261, 2262, 2263, 2264, 2265, 2266, 2267, 2268, 2269, 2270, 2271, 2272, 2273, 2274, 2275, 2276, 2277, 2278, 2279, 2280, 2281, 2282, 2283, 2284, 2285, 2286, 2287, 2288, 2289, 2290, 2291, 2292, 2293, 2294, 2295, 2296, 2297, 2298, 2299, 2300, 2301, 2302, 2303, 2304, 2305, 2306, 2307, 2308, 2309, 2310, 2311, 2312, 2313, 2314, 2315, 2316, 2317, 2318, 2319, 2320, 2321, 2322, 2323, 2324, 2325, 2326, 2327, 2328, 2329, 2330, 2331, 2332, 2333, 2334, 2335, 2336, 2337, 2338, 2339, 2340, 2341, 2342, 2343, 2344, 2345, 2346, 2347, 2348, 2349, 2350, 2351, 2352, 2353, 2354, 2355, 2356, 2357, 2358, 2359, 2360, 2361, 2362, 2363, 2364, 2365, 2366, 2367, 2368, 2369, 2370, 2371, 2372, 2373, 2374, 2375, 2376, 2377, 2378, 2379, 2380, 2381, 2382, 2383, 2384, 2385, 2386, 2387, 2388, 2389, 2390, 2391, 2392, 2393, 2394, 2395, 2396, 2397, 2398, 2399, 2400, 2401, 2402, 2403, 2404, 2405, 2406, 2407, 2408, 2409, 2410, 2411, 2412, 2413, 2414, 2415, 2416, 2417, 2418, 2419, 2420, 2421, 2422, 2423, 2424, 2425, 2426, 2427, 2428, 2429, 2430, 2431, 2432, 2433, 2434, 2435, 2436, 2437, 2438, 2439, 2440, 2441, 2442, 2443, 2444, 2445, 2446, 2447, 2448, 2449, 2450, 2451, 2452, 2453, 2454, 2455, 2456, 2457, 2458, 2459, 2460, 2461, 2462, 2463, 2464, 2465, 2466, 2467, 2468, 2469, 2470, 2471, 2472, 2473, 2474, 2475, 2476, 2477, 2478, 2479, 2480, 2481, 2482, 2483, 2484, 2485, 2486, 2487, 2488, 2489, 2490, 2491, 2492, 2493, 2494, 2495, 2496, 2497, 2498, 2499, 2500, 2501, 2502, 2503, 2504, 2505, 2506, 2507, 2508, 2509, 2510, 2511, 2512, 2513, 2514, 2515, 2516, 2517, 2518, 2519, 2520, 2521, 2522, 2523, 2524, 2525, 2526, 2527, 2528, 2529, 2530, 2531, 2532, 2533, 2534, 2535, 2536, 2537, 2538, 2539, 2540, 2541, 2542, 2543, 2544, 2545, 2546, 2547, 2548, 2549, 2550, 2551, 2552, 2553, 2554, 2555, 2556, 2557, 2558, 2559, 2560, 2561, 2562, 2563, 2564, 2565, 2566, 2567, 2568, 2569, 2570, 2571, 2572, 2573, 2574, 2575, 2576, 2577, 2578, 2579, 2580, 2581, 2582, 2583, 2584, 2585, 2586, 2587, 2588, 2589, 2590, 2591, 2592, 2593, 2594, 2595, 2596, 2597, 2598, 2599, 2600, 2601, 2602, 2603, 2604, 2605, 2606, 2607, 2608, 2609, 2610, 2611, 2612, 2613, 2614, 2615, 2616, 2617, 2618, 2619, 2620, 2621, 2622, 2623, 2624, 2625, 2626, 2627, 2628, 2629, 2630, 2631, 2632, 2633, 2634, 2635, 2636, 2637, 2638, 2639, 2640, 2641, 2642, 2643, 2644, 2645, 2646, 2647, 2648, 2649, 2650, 2651, 2652, 2653, 2654, 2655, 2656, 2657, 2658, 2659, 2660, 2661, 2662, 2663, 2664, 2665, 2666, 2667, 2668, 2669, 2670, 2671, 2672, 2673, 2674, 2675, 2676, 2677, 2678, 2679, 26

input: "130"



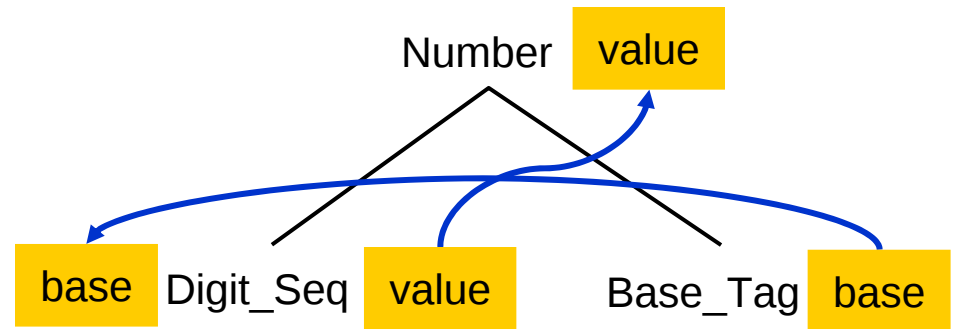
Attribute evaluation by tree walking



- at each node:
 - try to perform all the assignments in the evaluation rules for that node
 - visit all children
 - again try to perform the attribute assignments
- repeat walking until top-level attributes are assigned

Attribute evaluation by tree walking

```
walk number(node)
  evaluate number(node)
  walk digit_seq(node.digit_seq)
  walk base_tag(node.base_tag)
  evaluate number(node)
```



```
evaluate number(node)
  IF node.value is not set AND node.digit_seq.value is set THEN
    SET node.value TO node.digit_seq.value
  IF node.digit_seq.base is not set AND node.base_tag.base is set THEN
    SET node.digit_seq.base TO node.base_tag.base
```

Exercise (6 min.)



WHILE Number.value is not set:
 walk number(Number);

- how many tree walks are necessary to evaluate the attributes of the AST representing the octal number '130' ?
- how many for '1234D' ?

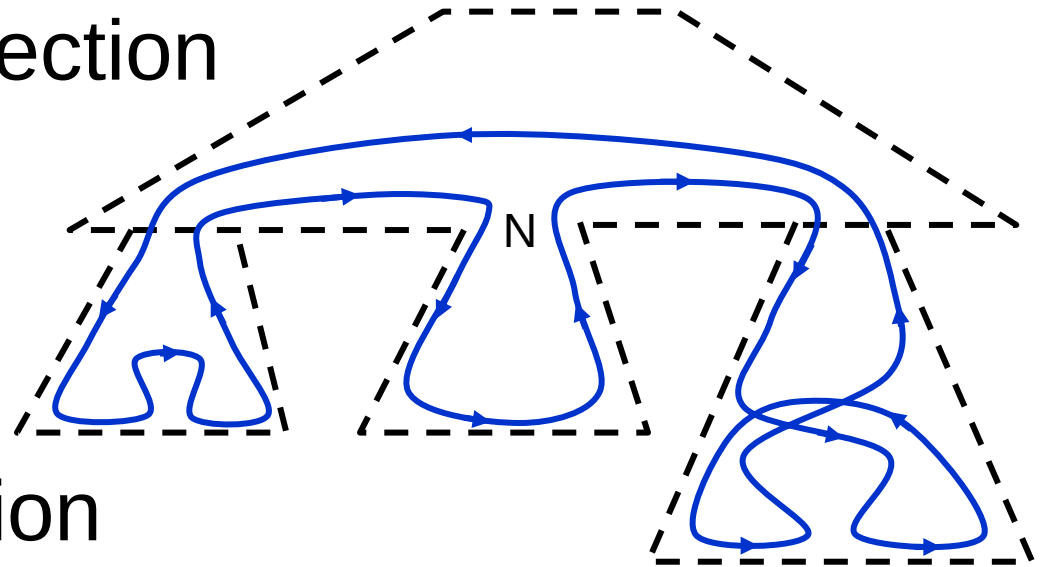
Answers

A thick, horizontal yellow brushstroke underline that spans most of the width of the slide, positioned directly beneath the title.

Cycle handling

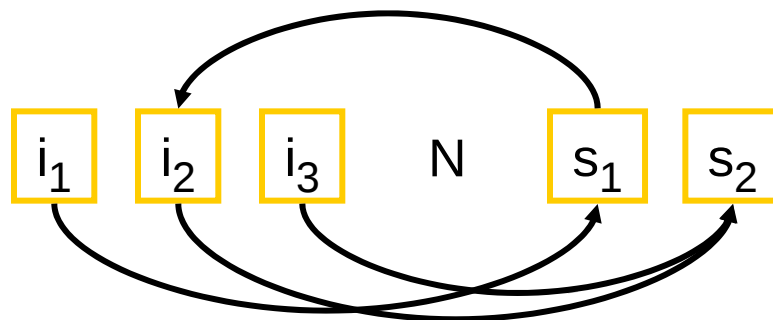
- **dynamic** cycle detection

$\#walks > \#attributes$



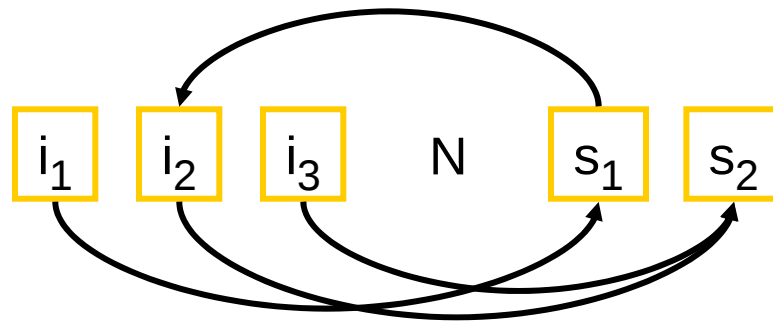
- **static** cycle detection

transitive closure of IS-SI graphs, see book



IS-SI graphs

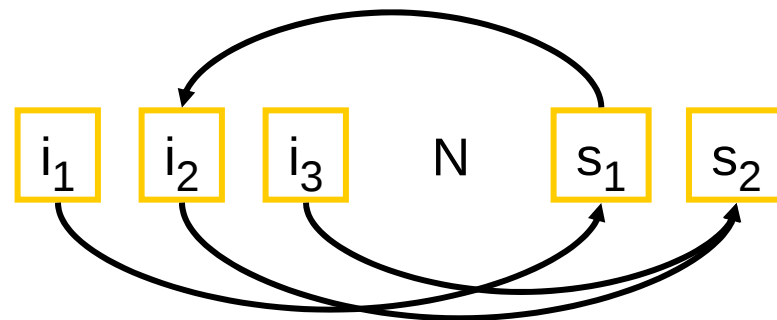
Synthesized-Inherited dependencies capture the dependencies in all trees **above** N



Inherited-Synthesized dependencies capture the dependencies in all trees **below** N

Multi-visit attribute grammars

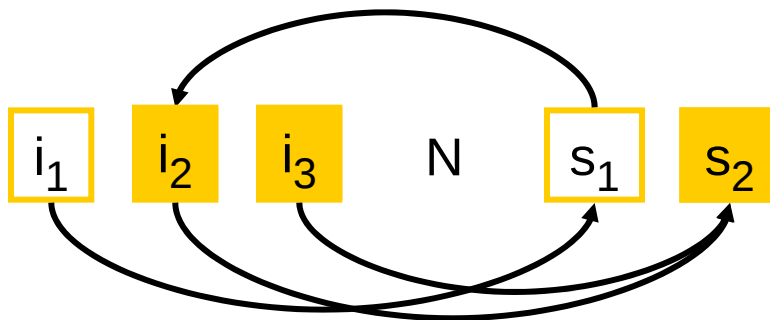
- avoid interpretation overhead
- generate code for each visit that “knows” what attributes to assign and which children to visit



- static attribute partition: $(IN_i, SN_i)_{i=1..n}$
 $(\{i_1, i_3\}, \{s_1\}), (\{i_2\}, \{s_2\})$

Ordered attribute grammars

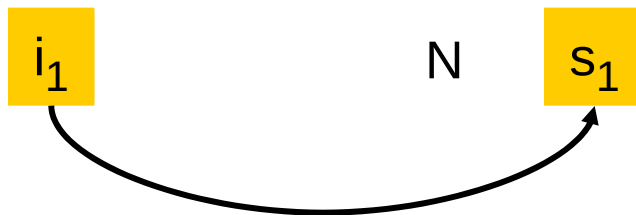
- late evaluation partitioning heuristic
 - work backwards
 - find (IN_{last}, SN_{last}) of the last visit
 - SN_{last} has no outgoing edges in IS-SI graph
 - IN_{last} is the set of attributes SN_{last} depends on
 - remove IN_{last} and SN_{last} from the IS-SI graph
 - repeat until the IS-SI graph is empty



$(\{i_2, i_3\}, \{s_2\})$

Ordered attribute grammars

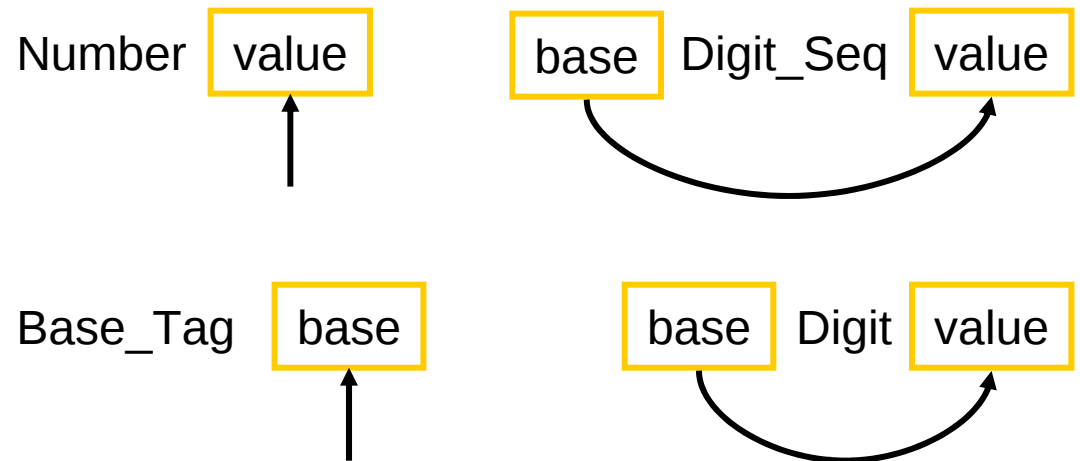
- late evaluation partitioning heuristic
 - work backwards
 - find (IN_{last}, SN_{last}) of the last visit
 - SN_{last} has no outgoing edges in IS-SI graph
 - IN_{last} is the set of attributes SN_{last} depends on
 - remove IN_{last} and SN_{last} from the IS-SI graph
 - repeat until the IS-SI graph is empty



$(\{i_1\}, \{s_1\})$ $(\{i_2, i_3\}, \{s_2\})$

Example: integral numbers

- IS-SI graphs:



- attribute partitioning:
(late evaluation)

	IN ₁	SN ₁
Number		value
Digit_Seq	base	value
Digit	base	value
Base_Tag		base

Example: integral numbers

- attribute partitioning:
(late evaluation)

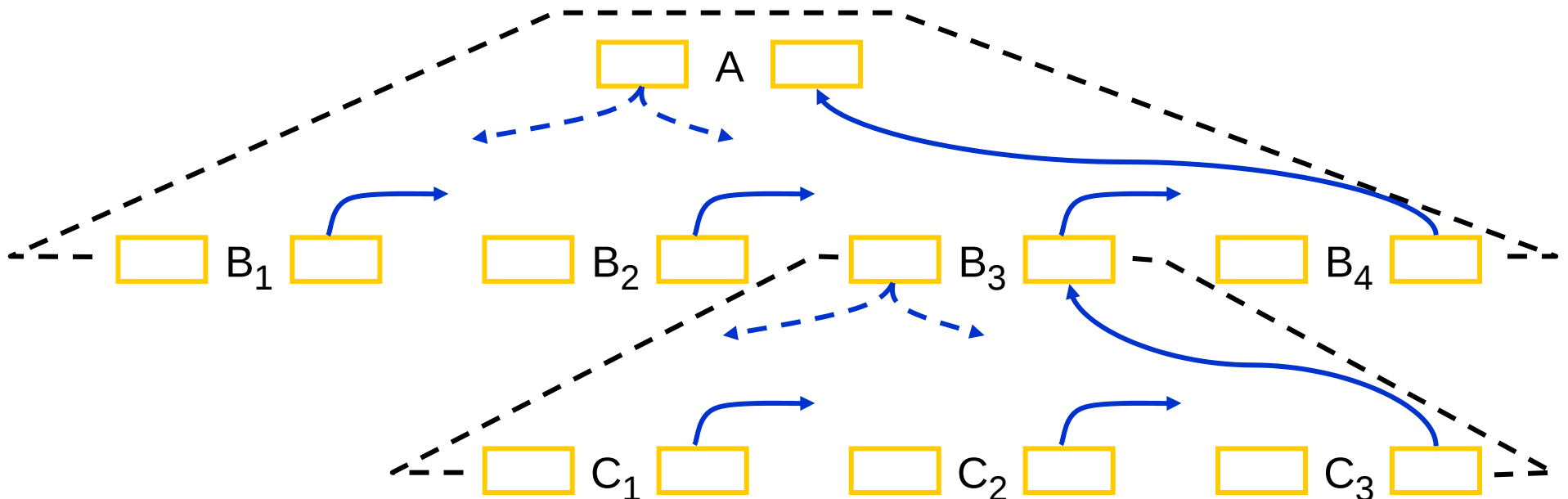
	IN_1	SN_1
Number		value
Digit_Seq	base	value
Digit	base	value
Base_Tag		base

- code:

```
visit_1 number(node)
  visit_1 base_tag(node.base_tag)
  SET node.digit_seq.base TO node.base_tag.base
  visit_1 digit_seq(node.digit_seq)
  SET node.value TO node.digit_seq.value
```

L-attributed grammars

- combine attribute grammars and **top-down** parsing
- attributes may only depend on information from the parent node or siblings to the left



LLnextgen example

```
main      : line+
          ;

line
  : expr '\n'      { printf("%d\n", expr); }
  ;

expr<int>
  : term           { LLretval = term; }
  [ '+' term<right> { LLretval += right; }
  ]*
  ;

term<int>
  : factor         { LLretval = factor; }
  [ '*' factor     { LLretval *= factor; }
  ]*
  ;

factor<int>
  : '(' expr ')'   { LLretval = expr; }
  | DIGIT          { LLretval = yylval; }
  ;
```

attributes

evaluation rules

Bottom-up parsing and attribute grammars

- stack of attributes
- problem with inherited attributes

$A \rightarrow B \{C.inh_attr := f(B.syn_attr); \} C$

code can only be executed at the end

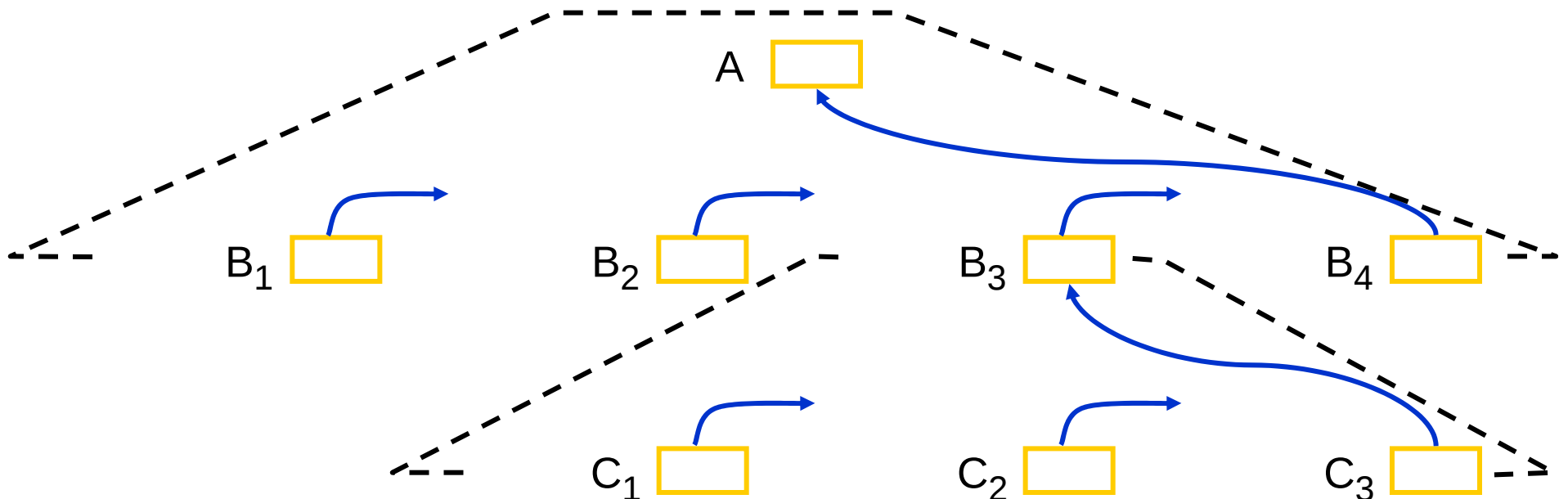
- solution: ϵ -rules

$A \rightarrow B \text{ Action1 } C$

$\text{Action1} \rightarrow \epsilon \{C.inh_attr := f(B.syn_attr); \}$

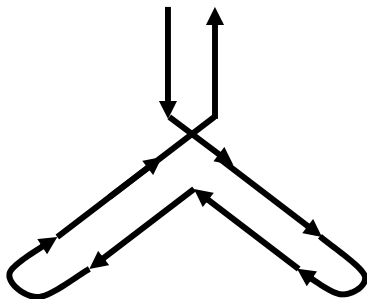
S-attributed grammars

- combine attribute grammars and **bottom-up** parsing
- only synthesized attributes may be used

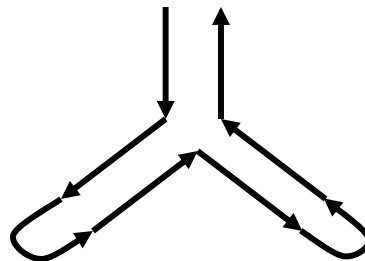


Summary

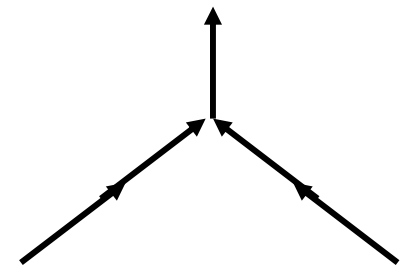
- attribute grammars
 - formal attributes per symbol: **inherited** & **synthesized**
 - attribute evaluation rule per production
- dependency graphs
- cycle detection
- evaluation order



multi-visit grammars



L -attributed grammars



S -attributed grammars