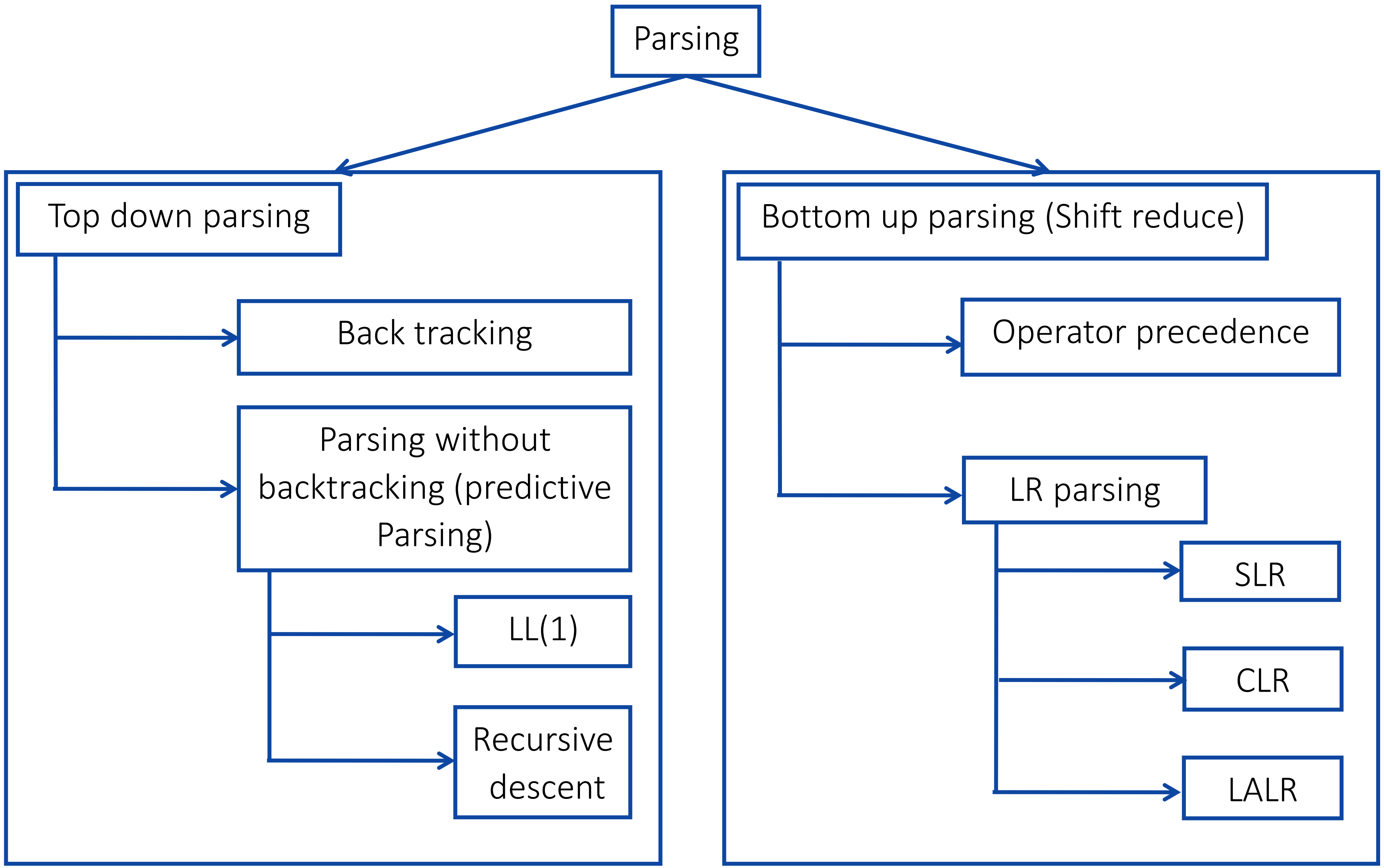
Module 2 – Syntax Analysis
Bottom Up Parsing

Parsing Methods



Handle & Handle pruning

- **Handle**: A "handle" of a string is a substring of the string that matches the right side of a production, and whose reduction to the non terminal of the production is one step along the reverse of rightmost derivation.
- **Handle pruning:** The process of discovering a handle and reducing it to appropriate left hand side non terminal is known as handle pruning.

E→E+E
E→E*E String: id1+id2*id3
E→id

Rightmost Derivation

E
E+E
E+E*E
E+E*id3
E+id2*id3
id1+id2*id3

Right sentential form	Handle	Production
id1+id2*id3		

Shift reduce parser

- The shift reduce parser performs following basic operations:

1. **Shift**: Moving of the symbols from input buffer onto the stack, this action is called shift.
2. **Reduce**: If handle appears on the top of the stack then reduction of it by appropriate rule is done. This action is called reduce action.
3. **Accept**: If stack contains start symbol only and input buffer is empty at the same time then that action is called accept.
4. **Error**: A situation in which parser cannot either shift or reduce the symbols, it cannot even perform accept action then it is called error action.

Example: Shift reduce parser

Grammar:

$E \rightarrow E+T \mid T$

$T \rightarrow T*F \mid F$

$F \rightarrow id$

String: id+id*id

Consider the following grammar-

$S \rightarrow T\ L\ ;$

$T \rightarrow int \mid float$

$L \rightarrow L\ ,\ id \mid id$

Parse the input string int id , id ; using a shift-reduce parser.

Operator precedence parsing

- **Operator Grammar**: A Grammar in which there is no Є in RHS of any production or no adjacent non terminals is called operator grammar.
- Example: E→ EAE | (E) | id

  A→ + | * | -

- Above grammar is not operator grammar because right side *EAE* has consecutive non terminals.
- In operator precedence parsing we define following disjoint relations:

Relation	Meaning
a<·b	a “yields precedence to” b
a=b	a “has the same precedence as” b
a·>b	a “takes precedence over” b

Precedence & associativity of operators

Operator	Precedence	Associative
↑	1	right
*, /	2	left
+, -	3	left

Steps of operator precedence parsing

1. Find Leading and trailing of non terminal
2. Establish relation
3. Creation of table
4. Parse the string

Leading & Trailing

Leading:- Leading of a non terminal is the first terminal or operator in production of that non terminal.

Trailing:- Trailing of a non terminal is the last terminal or operator in production of that non terminal.

Example: $E \rightarrow E+T \mid T$

$T \rightarrow T*F \mid F$

$F \rightarrow id$

Non terminal	Leading	Trailing
E		
T		
F		

Rules to establish a relation

1. For a = b, $\Rightarrow$ aAb, where A is ϵ or a single non terminal [e.g : (E)]
2. a <·b $\Rightarrow$ $Op\,.NT\ \ then\ \ Op\ \ <.Leading(NT)$ [e.g : +T]
3. a ·>b $\Rightarrow$ $NT\,.Op\ then\ (Trailing(NT))\cdot > Op$ [e.g : E+]
4. $ <· Leading (start symbol)
5. Trailing (start symbol) ·> $

Example: Operator precedence parsing

Step 1: Find Leading & Trailing of NT

Nonterminal	Leading	Trailing
E	{+,*,id}	{+,*,id}
T	{*,id}	{*,id}
F	{id}	{id}

$E \rightarrow E+T \mid T$
$T \rightarrow T*F \mid F$
$F \rightarrow id$

Step 2: Establish Relation

a <·b

$Op \cdot NT$	$Op \lessdot Leading(NT)$
$+T$	$+ \lessdot \{*, id\}$
$*F$	$* \lessdot \{id\}$

Step3: Creation of Table

	+	*	id	$
+				
*				
id				
$				

Example: Operator precedence parsing

Step 1: Find Leading & Trailing of NT

Nonterminal	Leading	Trailing
E	{+,*,id}	{+,*,id}
T	{*,id}	{*,id}
F	{id}	{id}

E→ E+ T| T
T→ T* F| F
F→ id

Step2: Establish Relation

$a \cdot> b$

$$NT \cdot Op \quad \Big| \quad (Trailing(NT)) \cdot> Op$$

$$E + \quad \Big| \quad \{+,*,id\} \cdot> +$$

$$T * \quad \quad \{*,id\} \cdot> *$$

Step3: Creation of Table

	+	*	id	$
+		$<\cdot$	$<\cdot$	
*			$<\cdot$	
id				
$				

Example: Operator precedence parsing

Step 1: Find Leading & Trailing of NT

Nonterminal	Leading	Trailing
E	{+,*,id}	{+,*,id}
T	{*,id}	{*,id}
F	{id}	{id}

E→ E+ T| T
T→ T* F| F
F→ id

Step 2: Establish Relation

$<· Leading (start symbol)

$\$ <\cdot \{+, *, id\}$

Trailing (start symbol) ·> $

$\{+, *, id\} \cdot> \ \$$

Step 3: Creation of Table

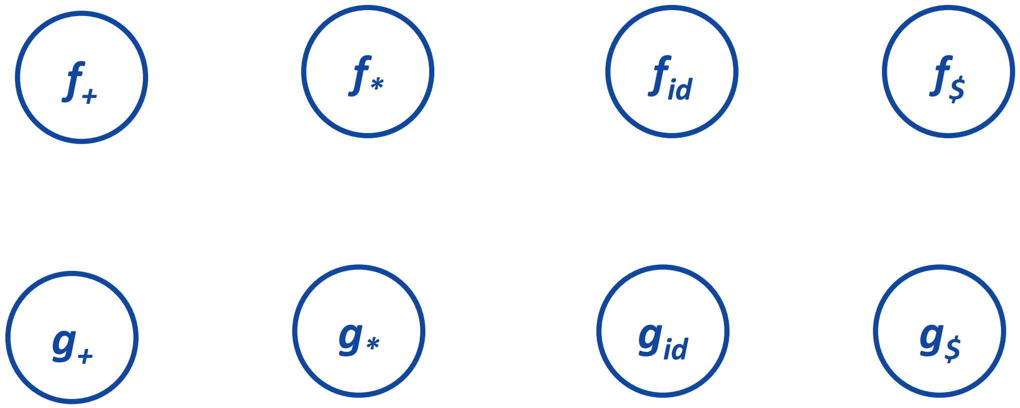
	+	*	id	$
+	·>	<·	<·	
*	·>	·>	<·	
id	·>	·>		
$				

Operator precedence function

1. Create functions f_a and g_a for each a that is terminal or \$.

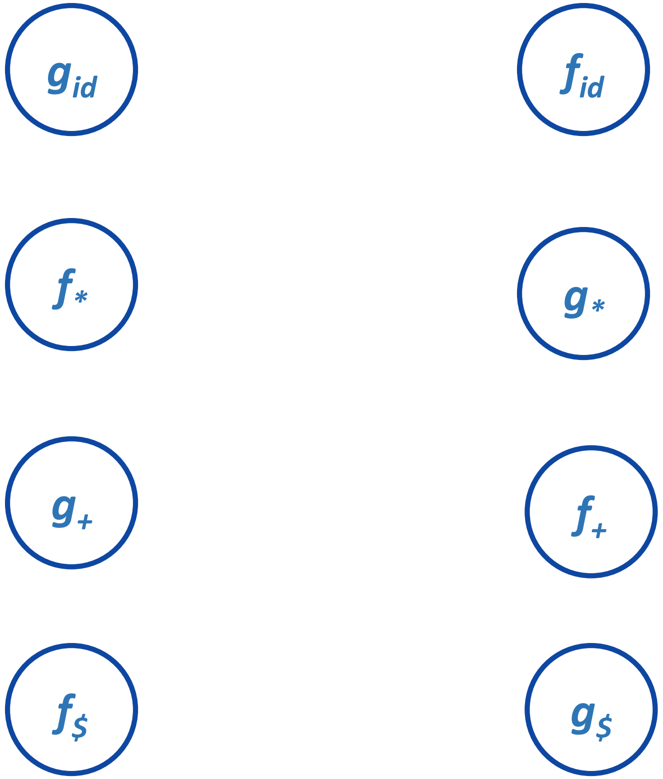
E→ E+T | T
T→ T*F | F
F→ id

$$a = \{+, *, id\} \ or \ \$$$



Operator precedence function

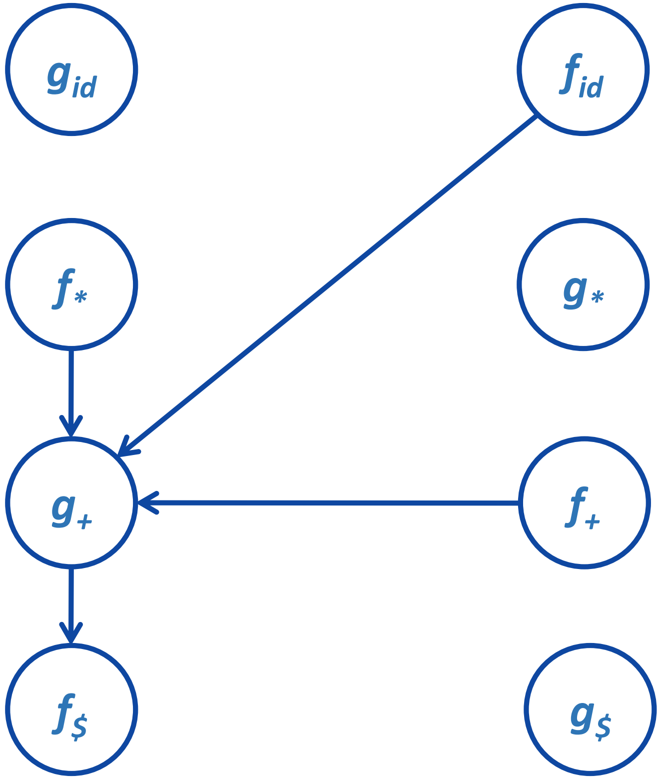
- *Partition the symbols in as many as groups possible, in such a way that f_a and g_b are in the same group if $a = b$.*



	+	*	id	$
+	·>	<·	<·	·>
*	·>	·>	<·	·>
id	·>	·>		·>
$	<·	<·	<·	

Operator precedence function

3. *if a <· b, place an edge from the group of g_b to the group of f_a*
   if a ·> b, place an edge from the group of f_a to the group of g_b

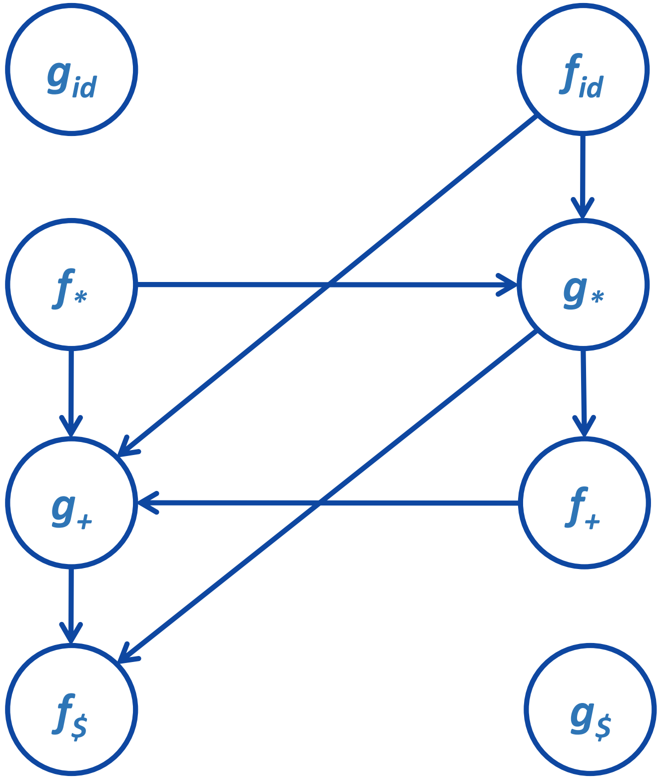


f \ g	+	*	id	$
+	·>	<·	<·	·>
*	·>	·>	<·	·>
id	·>	·>		·>
$	<·	<·	<·	

$f_+ \cdot\!> g_+$	$f_+ \rightarrow g_+$
$f_* \cdot\!> g_+$	$f_* \rightarrow g_+$
$f_{id} \cdot\!> g_+$	$f_{id} \rightarrow g_+$
$f_\$ <\!\cdot\ g_+$	$f_\$ \leftarrow g_+$

Operator precedence function

3. *if a <· b, place an edge from the group of g_b to the group of f_a*
   if a ·> b, place an edge from the group of f_a to the group of g_b



			g		
		+	*	id	$
f	+	·>	<·	<·	·>
	*	·>	·>	<·	·>
	id	·>	·>		·>
	$	<·	<·	<·	

$f_+ <\cdot\, g_*$ $\qquad f_+ \leftarrow g_*$

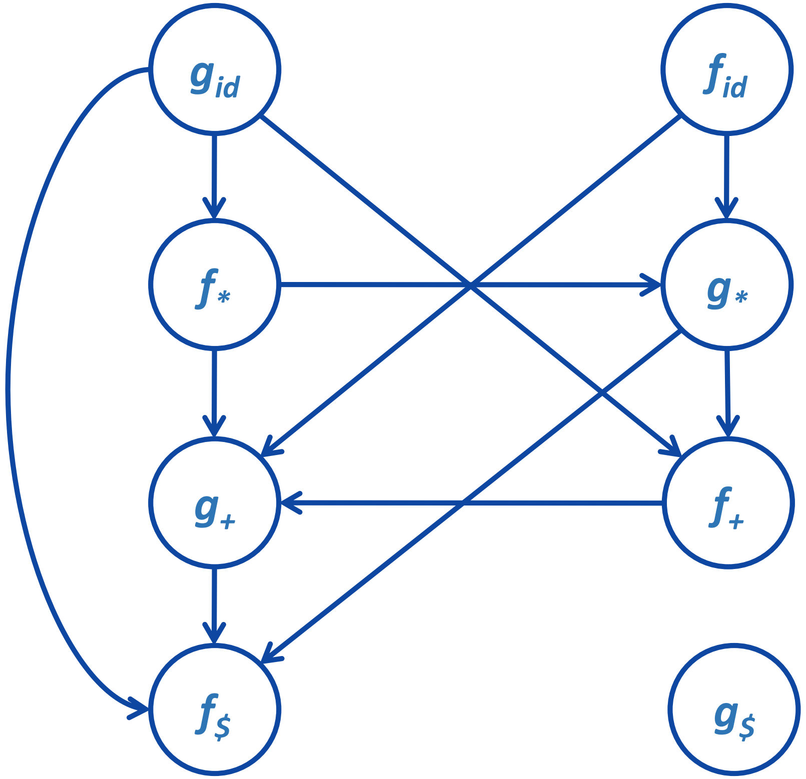
$f_* \cdot\!> g_*$ $\qquad f_* \rightarrow g_*$

$f_{id} \cdot\!> g_*$ $\qquad f_{id} \rightarrow g_*$

$f_\$ <\cdot\, g_*$ $\qquad f_\$ \leftarrow g_*$

Operator precedence function

3. *if a <· b, place an edge from the group of g_b to the group of f_a*
   if a ·> b, place an edge from the group of f_a to the group of g_b



f \ g	+	*	id	$
+	·>	<·	<·	·>
*	·>	·>	<·	·>
id	·>	·>		·>
$	<·	<·	<·	

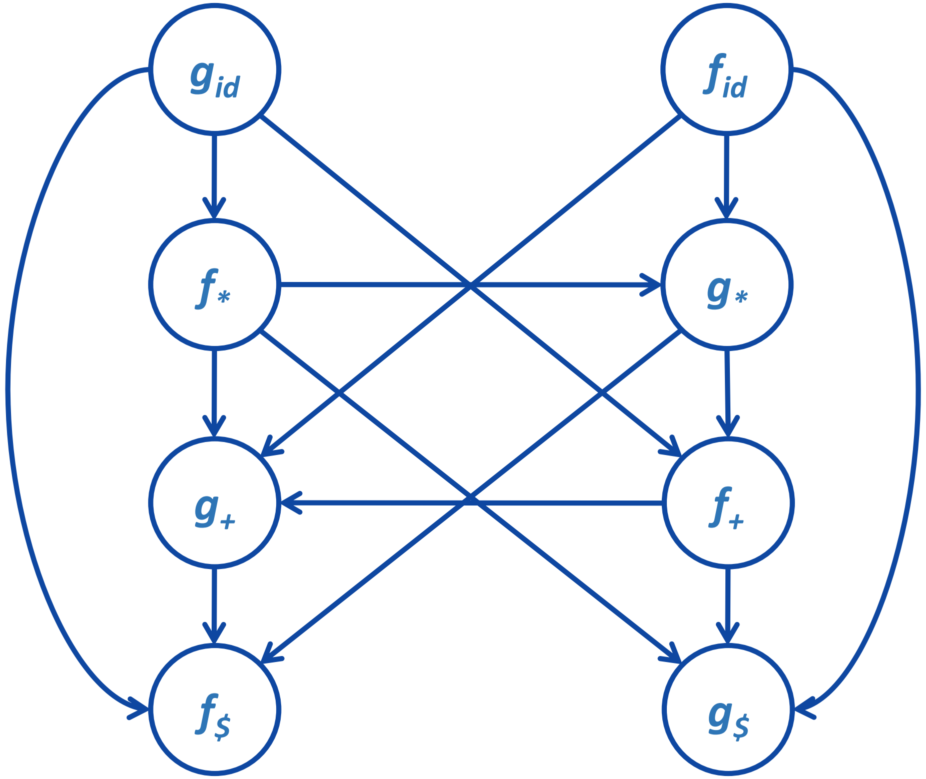
$f_+ <\cdot\, g_{id}$ $\quad f_+ \leftarrow g_{id}$

$f_* <\cdot\, g_{id}$ $\quad f_* \leftarrow g_{id}$

$f_\$ <\cdot\, g_{id}$ $\quad f_\$ \leftarrow g_{id}$

Operator precedence function

3. *if a <· b, place an edge from the group of g_b to the group of f_a*
   if a ·> b, place an edge from the group of f_a to the group of g_b



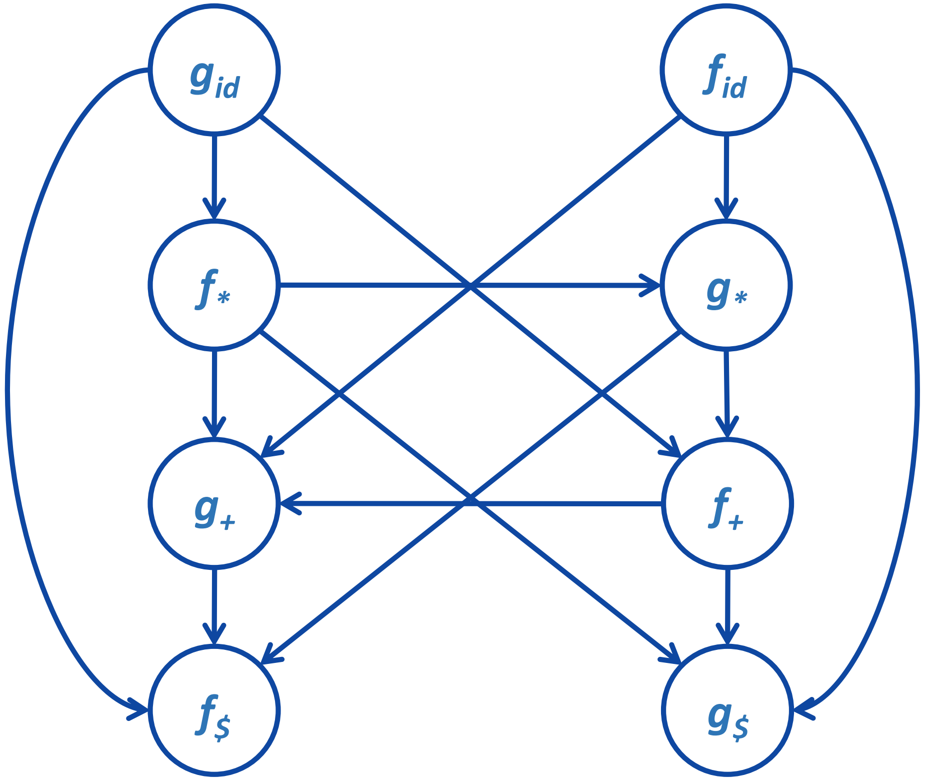
		g			
		+	*****	**id**	**$**
f	**+**	·>	<·	<·	·>
	*****	·>	·>	<·	·>
	id	·>	·>		·>
	$	<·	<·	<·	

$f_+ <\cdot\ g_\$$ $f_+ \rightarrow g_\$$

$f_* <\cdot\ g_\$$ $f_* \rightarrow g_\$$

$f_{id} <\cdot\ g_\$$ $f_{id} \rightarrow g_\$$

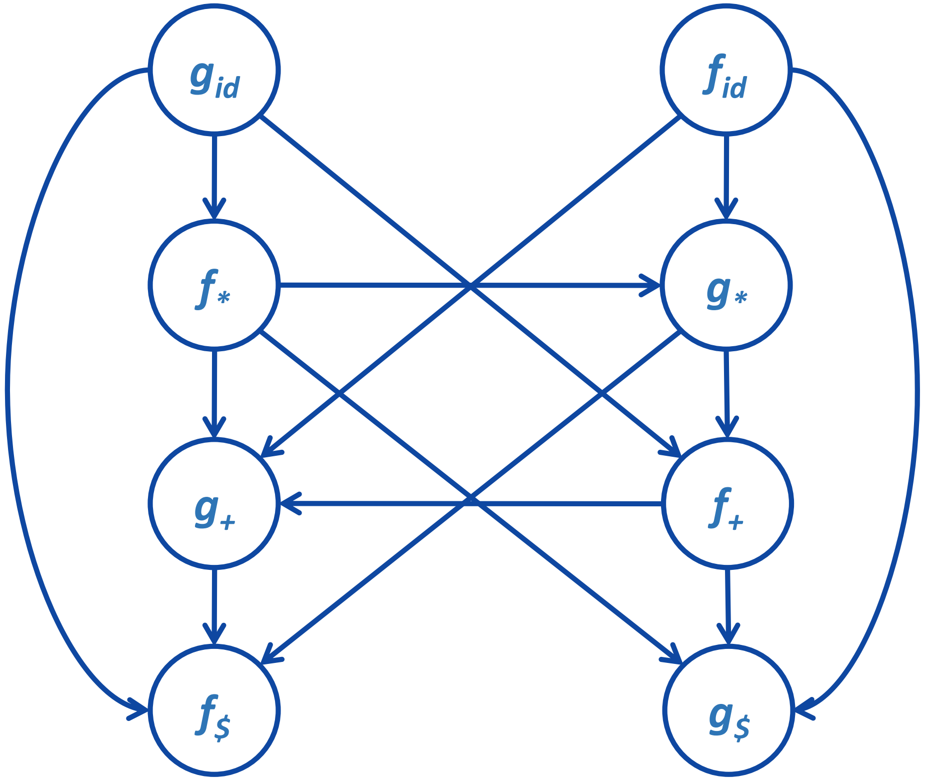
Operator precedence function



	+	*	id	$
f				
g				

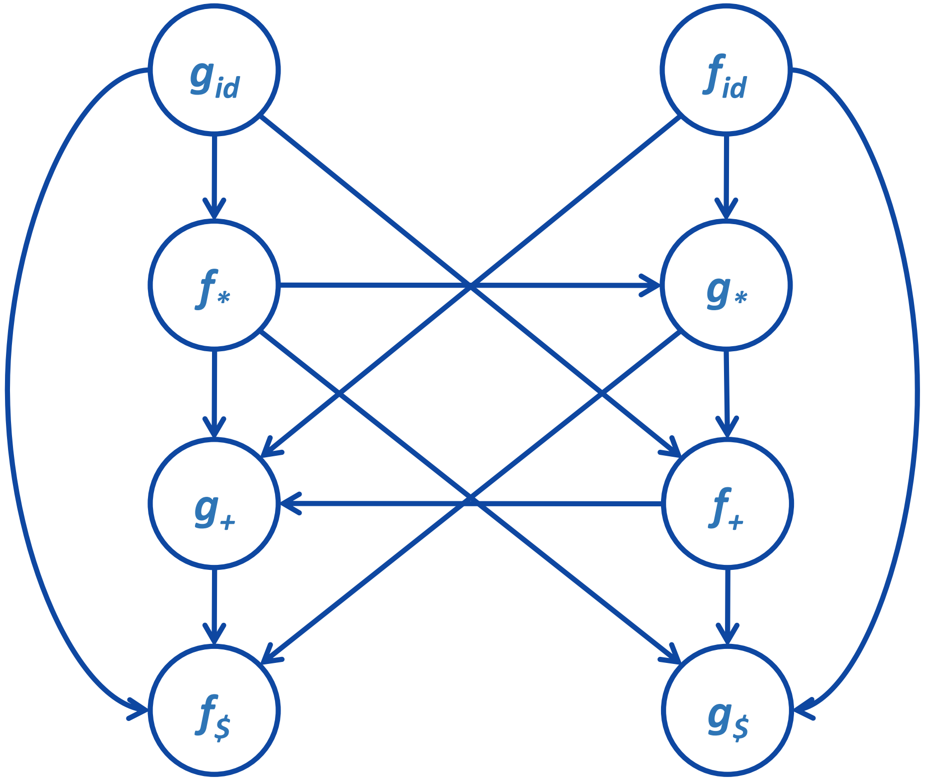
4. *If the constructed graph has a cycle then no precedence functions exist. When there ~~are no cycles collect the length of the longest paths from the groups of~~ ~~f_a and g_b respectively.~~*

Operator precedence function



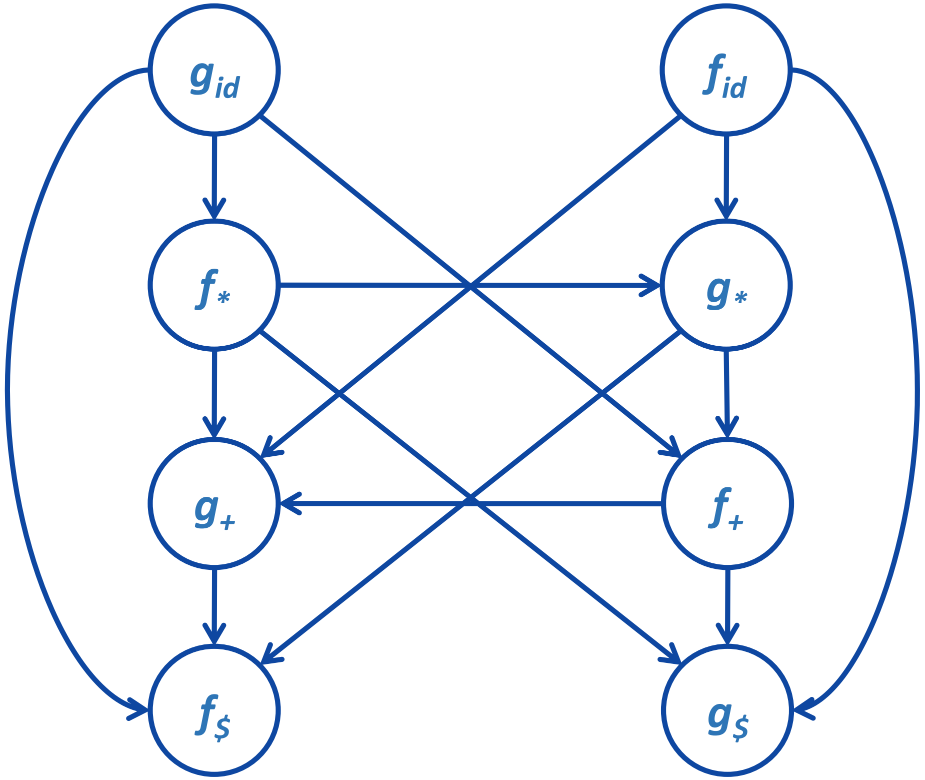
	+	*	id	$
f	2			
g				

Operator precedence function



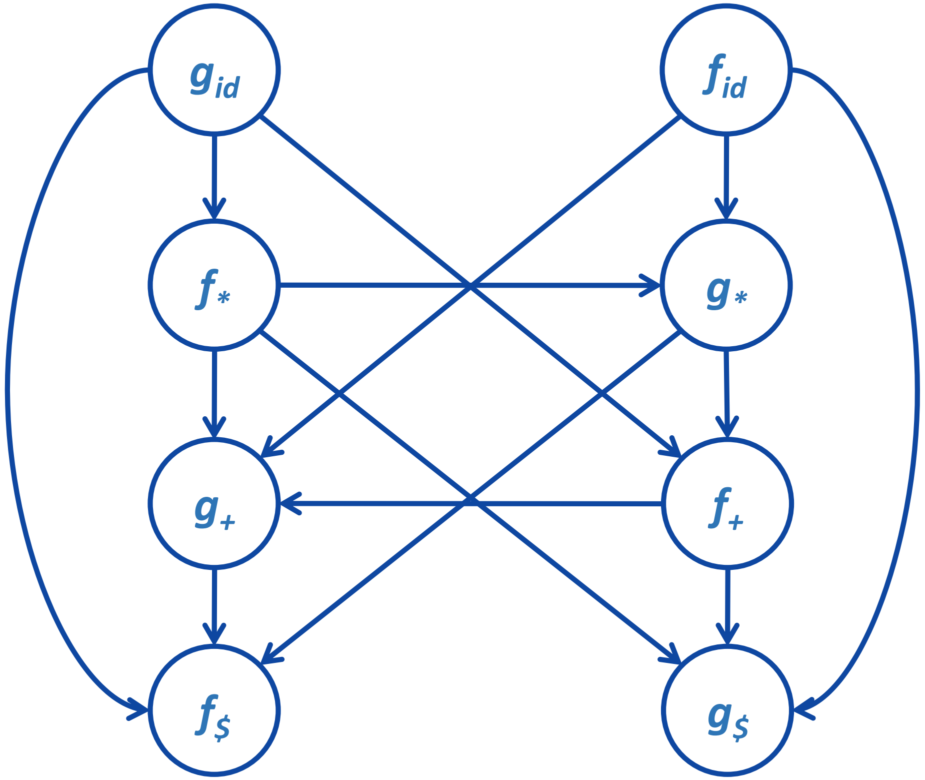
	+	*	id	$
f	2			
g	1			

Operator precedence function



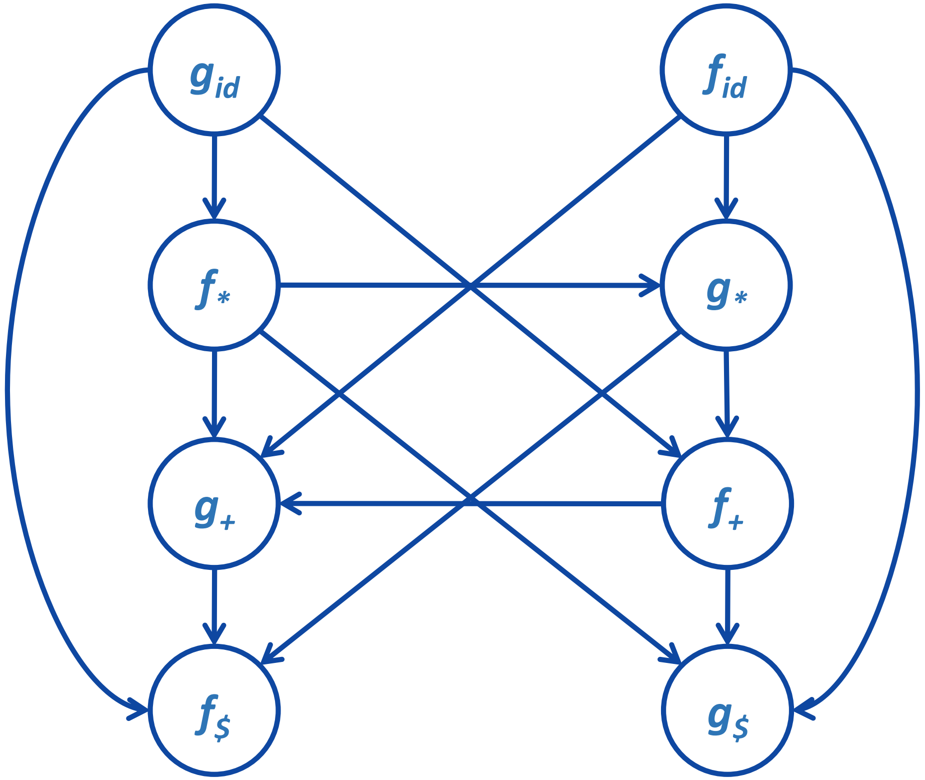
	+	*	id	$
f	2	4		
g	1			

Operator precedence function



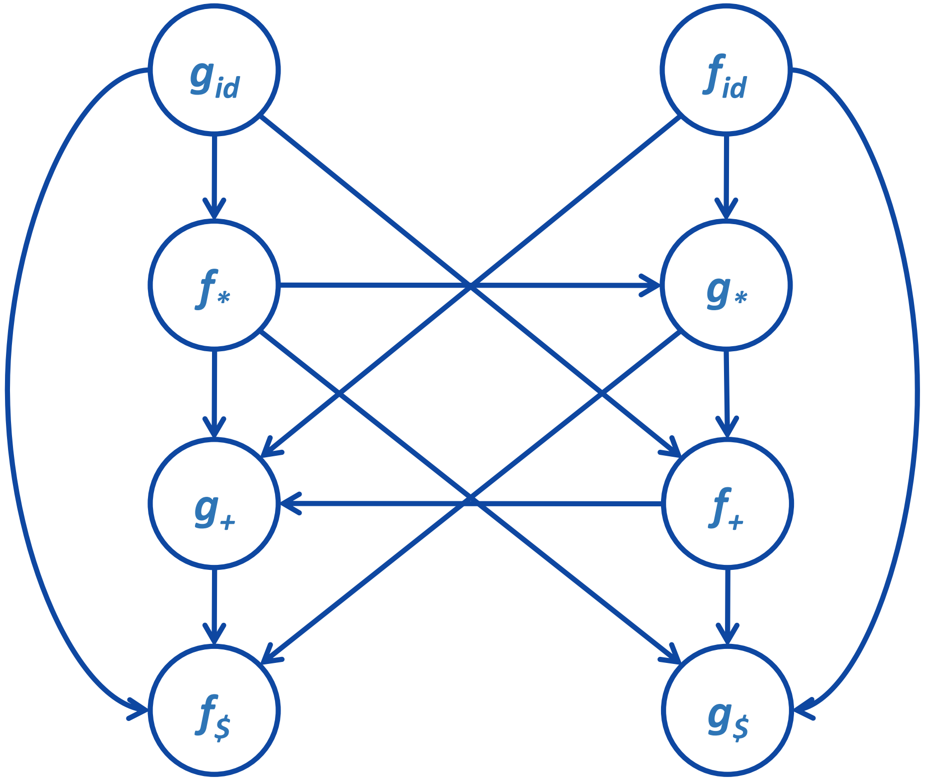
	+	*	id	$
f	2	4		
g	1	3		

Operator precedence function



	+	*	id	$
f	2	4	4	
g	1	3		

Operator precedence function



	+	*	id	$
f	2	4	4	
g	1	3	5	