A Factor Replacement System (FRS) is a finite set of pair of positive integers
$F = \{(a_1,b_1), \ldots (a_n,b_n)\}$. Each pair is called a rule and may be denoted as a fraction b_i/a_i or a grammar style rule $a_i x \rightarrow b_i x$ or as a pair as shown in the definition.

A configuration (ID), x, of a FRS is a positive integer.

A rule b_i/a_i is enabled by some ID x, if x is divisible by a_i.
Computation is defined by multiplying x by the fraction b_i/a_i.provided the rule is enabled by x.

We define derivation in F by $x \Rightarrow_F y$ iff $y = x \times b_i/a_i$.where x is divisible by a_i. .

The concept of derivation in zero or more steps, $\Rightarrow_F^*$ is then the reflexive, transitive closure of $\Rightarrow_F$. As usual, we omit the F if it is understood by context.

The derivation or word problem for F is then the problem to determine of two arbitrary positive integers x and y, whether or not $x \Rightarrow_F^* y$.

Ordered Factor Replacement Systems are merely FRS's where the rules are totally ordered and
$x \Rightarrow_F y$ iff $y = x \times b_i/a_i$.where x is divisible by a_i and x is not divisible by any a_j, $j<i$.

The halting problem for an ordered FRS F is the problem to determine of an arbitrary positive integer x whether or not there is some y such that $x \Rightarrow_F^* y$ and y is terminal, meaning that there is no z such that $y \Rightarrow_F z$. This means that y enables no rules. An alternative version says there is no z, $z \neq y$, such that $y \Rightarrow_F z$. This allows termination by reaching a fixed point, rather than having no applicable rule.

FRS's are not computationally complete, whereas ordered FRS's are.

The notion of computation with ordered FRS's is typically to start with inputs as the exponents of successive primes, starting with 3, e.g., if we call the 0-th prime 2, the first 3, etc, then $3^{x1}5^{x2} \ldots p_n^{xn}$ represents a start with input x1, x2, …, xn. When termination occurs, we have the answer as the exponent of $p_0 = 2$. That is, for function f, FRS F computes f, if, for all x1, x2, …, xn,
$3^{x1}5^{x2} \ldots p_n^{xn} \Rightarrow_F^* p_0^{f(x1, x2, \ldots, xn)} Z$. where Z contains no factors of 2.
Moreover, the last ID above is terminal (or a fixed point).

A Factor Replacement System with Residue is a finite set of quads of positive integers $F = \{(a_1,b_1,c_1,d_1), \ldots (a_n,b_n,c_n,d_n)\}$. Each quad is called a rule and may be denoted as a a grammar style rule $a_ix + b_i \rightarrow c_ix + d_i$ or as a quad as shown in the definition.

A configuration (ID), x, of a FRS is a positive integer.

A rule $a_ix + b_i$ is enabled by some ID w, if $w = a_ix + b_i$ for some positive integer x.

We define derivation in F by $w \Rightarrow_F y$ iff $y = c_ix + d_i$ where $w = a_ix + b_i$.

The concept of derivation in zero or more steps, $\Rightarrow_F^*$ is then the reflexive, transitive closure of $\Rightarrow_F$. As usual, we omit the F if it is understood by context.

The derivation or word problem for F is then the problem to determine of two arbitrary positive integers x and y, whether or not $x \Rightarrow_F^* y$.

There is no need for a notion of ordered FRS's with residue as the unordered variety are complete computational devices.

The halting problem for an FRS with residue F is the problem to determine of an arbitrary positive integer w whether or not there is some y such that $w \Rightarrow_F^* y$ and y is terminal, meaning that there is no z such that $y \Rightarrow_F z$. This means that y enables no rules. An alternative version says there is no z, $z \neq y$, such that $y \Rightarrow_F z$. This allows termination by reaching a fixed point, rather than having no applicable rule.

The notion of computation with FRS's with Residue is typically to start with inputs as the exponents of successive primes, starting with 3, e.g., if we call the 0-th prime 2, the first 3, etc, then $3^{x1}5^{x2} \ldots p_n^{xn}$ represents a start with input x1, x2, …, xn. When termination occurs, we have the answer as the exponent of $p_0 = 2$. That is, for function f, FRS F computes f, if, for all x1, x2, …, xn,
$3^{x1}5^{x2} \ldots p_n^{xn} \Rightarrow_F^* p_0^{f(x1, x2, \ldots, xn)} Z$. where Z contains no factors of 2.
Moreover, the last ID above is terminal (or a fixed point). Of course, these can compute multiple values, so we restrict ourselves to FRS's with residue that have no overlapping rules. That is, for $i \neq j$, there is no w that enables both i and j.

A Petri Net is a 4-tuple P = (S,T,F,W) where

S is a finite, non-empty set of places or nodes;
T is a finite, non-empty set of transitions;
$S \cap T = \varnothing$
$F \subseteq S \times T \cup T \times S$ is the flow relation, one associated with each transition;
$W: S \times T \cup T \times S \rightarrow \aleph$ is the weight function, where $W(x,y) = 0$ iff $(x,y) \notin F$

A configuration (ID), M, of a Petri Net, aka a marking, is a point in non-negative integer |S|-space. The i-th element of the marking vector, M(i), specifies the number of markers in the corresponding place.

Computation is defined by the firings of transitions.
A transition $t \in T$ is enabled by some marking M, denoted M[t> if
$W(s,t) \geq M(s)$ for all $s \in S$.

If $t \in T$ is enabled by M then t may fire. If it fires, then M is changed to M', where
$M'(s) = M(s) - W(s,t) + W(t,s)$ for all $s \in S$.

We denote a single step derivation by firing t as M [t> M'.

We define derivation in P by $M \Rightarrow_P M'$ iff M [t> M', for some $t \in T$.

The concept of derivation in zero or more steps, $\Rightarrow_P^*$ is then the reflexive, transitive closure of $\Rightarrow_P$. As usual, we omit the P if it is understood by context.

The derivation or word problem for P is then the problem to determine of two arbitrary markings M and M', whether or not $M \Rightarrow_P^* M'$.

Ordered Petri Nets are merely Petri Nets where the transitions are totally ordered and M [t> M' iff t is the least numbered transition enabled by marking M.

The halting problem for an ordered Petri Net P is the problem to determine of an arbitrary marking M whether or not there is some M' such that $M \Rightarrow_P^* M'$ and M' is terminal, meaning that there is no M'' such that $M' \Rightarrow_P M''$. This means that M' enables no transition.

For normal (unordered transitions) Petri Nets, we are often interested in knowing if any firing sequence will lead to a deadlock, defined as a terminal marking. It is important to see that this is a different problem than halting since there can be many marking sequences in a normal Petri Net, but only one such sequence in an ordered Petri Net.

Petri Nets are not computationally complete, whereas ordered Petri Nets are.

The notion of computation with ordered Petri Nets is typically to start with inputs, x1, x2, …, xn, as the contents of successive nodes, starting with node 1, and assuming a node n+1 that will contain the answer. That is, for function f, Petri Net P computes f, if, for all x1, x2, …, xn,
$< x1, x2, \ldots, xn, 0, \ldots, 0> \Rightarrow_P^* <?, ?, \ldots, ?, f(x1, x2, \ldots, xn), ?, \ldots, ?>$
Moreover, this last ID is terminal.

A Vector Addition System (VAS) is a 4-tuple V = (n, R) where

n is a positive integer
R is a finite, non-empty set of rule vectors in integer n-space

A configuration (ID), u, of a VAS, is a point in non-negative integer n-space.

Computation is defined by the application of rule vectors.
A rule vector $r = \langle r_1, \ldots, r_n \rangle \in R$ is enabled by some point $u = \langle u_1, \ldots, u_n \rangle$, denoted $u[r\rangle$ if $u_i \geq |r_i|$, for all $r_i < 0$.

If $r \in R$ is enabled by u then r may be added to u. If it is, then u is changed to w, where $w = u + r$.

We denote a single step derivation by firing r as $u\ [r\rangle\ w$.

We define derivation in V by $u \Rightarrow_V w$ iff $u\ [r\rangle\ w$, for some $r \in R$.

The concept of derivation in zero or more steps, $\Rightarrow_V^*$ is then the reflexive, transitive closure of $\Rightarrow_V$. As usual, we omit the V if it is understood by context.

The derivation or word problem for V is then the problem to determine of two arbitrary points in non-negative integer n-space u and w, whether or not $u \Rightarrow_V^* w$.

Ordered VAS's are merely VAS's where the rule vectors are totally ordered and
$u\ [r\rangle\ w$ iff r is the least numbered rule vector enabled by point u in non-negative integer n-space.

The halting problem for an ordered VAS V is the problem to determine of an arbitrary point u in non-negative integer n-space whether or not there is some w in non-negative integer n-space such that $u \Rightarrow_V^* w$ and w is terminal, meaning that there is no x such that $w \Rightarrow_P x$. This means that w enables no rule vector.

For normal (unordered rules) VAS's, we are often interested in knowing if any derivation sequence will lead to a deadlock, defined as a terminal point. It is important to see that this is a different problem than halting since there can be many alternate derivations in a normal VAS, but only one such sequence in an ordered VAS.

VAS's are not computationally complete, whereas ordered VAS's are.

The notion of computation with ordered VAS's is typically to start with inputs, x1, x2, …, xn, as the contents of successive coordinates, starting with the first, and assuming an n+1-st coordinate that will contain the answer. That is, for function f, VAS V computes f, if, for all x1, x2, …, xn,
$\langle x1, x2, \ldots, xn, 0, \ldots, 0 \rangle \Rightarrow_V^* \langle ?, ?, \ldots, ?, f(x1, x2, \ldots, xn), ?, \ldots, ? \rangle$
Moreover, this last ID is terminal.