On Relation of Linear Diophantine Equation Systems with Commutative Grammars

Dmitry Korzun

Petrozavodsk State University (Russia)

RuFiDiM — Third Russian Finnish Symposium
on Discrete Mathematics
September 15–18, 2014, Petrozavodsk, Russa

Linear Diophantine Equation Systems

Let $\mathbb{Z}$ and $\mathbb{Z}_+$ be the set of integers and nonnegative integers

Homogenous linear Diophantine equation system (n equations, m unknowns)

$$Ax = \mathbb{O}, \quad \text{where } A \in \mathbb{Z}^{n \times m}, \; x \in \mathbb{Z}_+^m. \tag{1}$$

Diophantine monoid $M_A = \{x \in \mathbb{Z}_+^m \mid Ax = \mathbb{O}\}$

The set $\mathcal{H}$ of all irreducible solutions to (1) is called Hilbert basis

- $\mathcal{H}$ is finite and unique
- "linear independence" is inappropriate here
- minimality: $\forall\, h \in \mathcal{H}$ there exists no $x \in M_A$ ($x \neq h$, $x \neq \mathbb{O}$) s.t. $x \leq h$
- $|\mathcal{H}|$ is not polynomial-bounded by the system input size

The general solution to (1) is $\quad x = \sum_{h \in \mathcal{H}} c_h h \;$ for some $\; c_h \in \mathbb{Z}_+$.

Non-unique decomposition for some $x \in M_A$

LDE Systems in Non-negative Form

Known technique: any $a \in \mathbb{Z}$ can be codified $a = a' - a''$ for $a', a'' \in \mathbb{Z}_+$

Primary representation: $\min\{a', a''\} = 0$
or, in other words, $a' = \max\{a, 0\}$ and $a'' = -\min\{a, 0\}$

Let $A = A' - A''$, where $A', A'' \in \mathbb{Z}_+^{n \times m}$. Then equivalent system to (1)

$$\sum_{j=1}^{m} a'_{ij} x_j = \sum_{j=1}^{m} a''_{ij} x_j, \quad i = 1, 2, \ldots, n, \tag{2}$$

where $\min\{a'_{ij}, a''_{ij}\} = 0$ for any i, j. Solutions $x \in \mathbb{Z}_+^m$

System (2) can be used for modeling

Application Areas for LDE Models

$$\sum_{j=1}^{m} a'_{ij} x_j = \sum_{j=1}^{m} a''_{ij} x_j, \quad i = 1, 2, \ldots, n, \tag{2}$$

- **Routing (in computer networks)**
- Structure discovery (in network traffic)
- Structural ranking (in networks)
- Loop parallelization (in array-processing programs)
- Memory control (caching strategies)
- Verification (for network protocols and parallel processes)
- Unification (automated deduction)

Algorithm Complexity Problems

$$\sum_{j=1}^{m} a'_{ij} x_j = \sum_{j=1}^{m} a''_{ij} x_j, \quad i = 1, 2, \ldots, n, \tag{2}$$

- Searching a solution x to a non-homogenous LDE system is NP-complete
- *Searching a solution x to (2) can be done in polynomial time* (find a rational solution and multiply to the common denominator)
- Deciding, given a solution x, if $x \in \mathcal{H}$ is coNP-complete
- Searching the Hilbert basis $\mathcal{H}$ employs enumerative algorithms, which take exponential time on n, m, and $||A||$
- Counting problem $|\mathcal{H}| =?$ is #P-complex and belongs to #NP

Analysis of particular cases of A''

Commutative Context-Free Grammars: Languages with Free Word Order

Given a finite alphabet $\mathit{\Pi}$, a commutative string (word) over $\mathit{\Pi}$ is $\{\pi^{a_\pi}\}_{\pi \in \mathit{\Pi}}$, where $a_\pi \in \mathbb{Z}_+$ is the number of occurrences of π.

- the order of symbols is ignored and α is a multiset of symbols
- $\mathit{\Pi}^*$, $\mathit{\Pi}^+ \rightsquigarrow \mathit{\Pi}^{\circledast}$, $\mathit{\Pi}^{\oplus}$
- Parikh mapping from $\mathit{\Pi}^*$ to $\mathbb{Z}_+^{|\mathit{\Pi}|}$:

$$\#[\alpha] = a \in \mathbb{Z}_+^{|\mathit{\Pi}|}, \qquad \#_\pi[\alpha] = a_\pi \in \mathbb{Z}_+ \text{ for } \pi \in \mathit{\Pi}$$

Denote also $\star[a] = \{\pi^{a_\pi}\}_{\pi \in \mathit{\Pi}}$

Commutative Context-Free (CCF)

CCF-Grammars: Definition

A commutative CF-grammar without a start symbol is a 3-tuple

$$G = (N, \Sigma, R)$$

- nonterminals N and terminals Σ are finite disjoint sets
- rules R are a finite subset of $N \times \Sigma^* N^*$, where each rule $r \in R$ is

$$u \to \tau\rho, \quad \text{where } u \in N, \tau \in \Sigma^*, \rho \in N^*$$

The derivation (parsing) problem for a given CCF-grammar G:

Deciding $\varkappa'\alpha \Rightarrow^* \varkappa''\beta$ for given $\varkappa'\alpha \in \Sigma^* N^*$ and $\varkappa''\beta \in \Sigma^* N^*$

Theorem 1 (Dung T. Huynh, 1983) *The problem is NP-complete for the case $u \Rightarrow^* \tau$, where $u \in N$ and $\tau \in \Sigma^*$.*

CCF-Grammars: The Case of Homogenous LDE System

Remove terminals Σ^* from G

Let $n = |N|$ and $m = |R|$
$R_u = \{r \in R \mid r = (u \to \rho_r), \rho_r = \star[(a_{vr})_{v \in N}]\}$ for $u \in N$

Construct the homogenous LDE system

$$\sum_{r \in R} a_{ur} x_r = \sum_{r \in R_u} x_r, \quad u \in N \tag{3}$$

Unknowns are interpreted as the number of rule applications in cycles:

$$x = \#[\alpha \Rightarrow^+ \alpha], \quad \text{where } \alpha \in N^+$$

$R_u \cap R_v = \varnothing \rightsquigarrow$ the right-hand side of (3) consists of partitioned unknowns

CCF-Grammars: Hilbert Basis

Derivation $\alpha \Rightarrow^* \alpha$ is a cycle for $\alpha \in N^+$

A cycle is simple if it does not contain a proper cycle

Theorem 2 (Korzun, 1997) *$x \in \mathcal{H}$ if and only if x corresponds to a simple cycle $u \Rightarrow^+ u$ for some $u \in N$*

Recall LDE systems (2) and (3):

$$\sum_{j=1}^{m} a'_{ij} x_j = \sum_{j=1}^{m} a''_{ij} x_j, \quad i = 1, 2, \ldots, n, \tag{2}$$

$$\sum_{r \in R} a_{ur} x_r = \sum_{r \in R_u} x_r, \quad u \in N \tag{3}$$

Hence, $Ax = E(R)x$ for specific $\{0, 1\}$ matrix $E(R)$

CCF-Grammars: LDE Systems Subclass

$Ax = E(R)x$

$$E(R) \rightsquigarrow \begin{pmatrix} \overbrace{1\dots1}^{m_1} & \overbrace{0\dots0}^{m_2} & \cdots & \overbrace{0\dots0}^{m_n} \\ 0\dots0 & 1\dots1 & \cdots & 0\dots0 \\ \vdots & \vdots & \ddots & \vdots \\ 0\dots0 & 0\dots0 & \cdots & 1\dots1 \end{pmatrix}, \quad m_1 + \cdots + m_n = m$$

Theorem 3 (Korzun, 1999) *Given a homogenous LDE system with unknowns x and Hilbert basis $\mathcal{H}$, one can construct the system*

$$A \begin{pmatrix} x \\ y \end{pmatrix} = E(R) \begin{pmatrix} x \\ y \end{pmatrix}$$

with unknowns $\begin{pmatrix} x \\ y \end{pmatrix}$ *and Hilbert basis $\mathcal{H}^c$ such that*

$$\mathcal{H} = \left\{ x \mid \begin{pmatrix} x \\ y \end{pmatrix} \in \mathcal{H}^c \right\}$$

Routing Problem (in computer networks)

Network of N nodes. Routing from s to d exploits intermediaries u

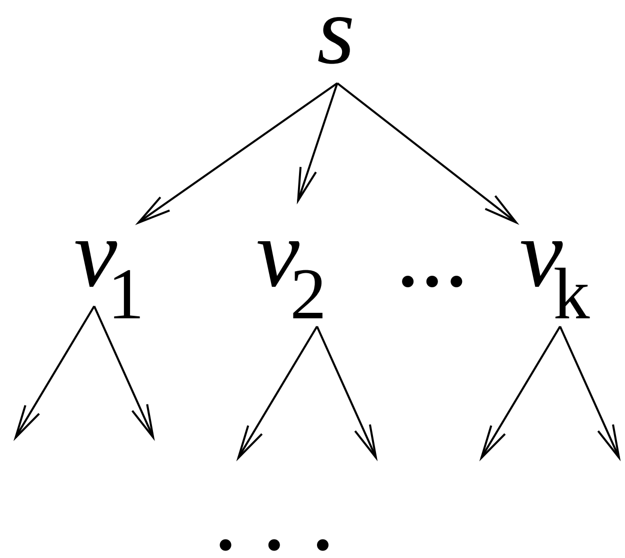
Routing table (neighbors): Node u keeps $T_u \subset N$ for direct communication

Let a packet targeted to d arrive at u

- *Base forwarding:* exactly one node v in T_u is selected
- *Retransmissions:* u sends to v,
  – if no ack then u retransmits ($\#\mathrm{attempt} = a_v$)
- *Sequential forwarding:* next-hop candidates v_1, v_2, $\ldots$, v_k,
  – initially, u sends to v_1
  – if no ack, sequentially to v_2, v_3, and so on up to v_k
  – $\#\mathrm{attempts} = a_i = a(v_i)$, $i = 1, 2, \ldots, k$
- *Parallel forwarding:* u sends simultaneously to v_1, v_2, $\ldots$, v_k
- *Path completion:* the packet is not forwarded further

Routing: Paths and Routes

- Network topology
  - digraph with outgoing links T_u; arcs $u \to v$
  - hypergraph: arc $\{u, v_1, \ldots, v_k\}$, weight $(a_1, \ldots, a_k)$
  - $u \to v_1^{a_1} v_2^{a_2} \cdots v_k^{a_k}$



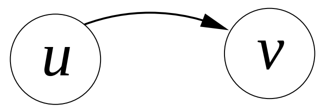
- When s sends a packet
  - there is a path in the digraph for each copy
  - atomic route is all paths
    $s \Rightarrow^* d_1^{b_1^+} \cdots d_k^{b_k^+}$ (not a tree in general)
- When $s_1, \ldots, s_l$ send $b_1^-, \ldots, b_l^-$ packets
  - aggregated route $s_1^{b_1^-} \cdots s_l^{b_l^-} \Rightarrow^* d_1^{b_1^+} \cdots d_k^{b_k^+}$
  - $s_1, \ldots, s_l$ act independently $\rightsquigarrow$ **context-free**

Routing Grammar: topology

A grammar rule describes a forwarding option

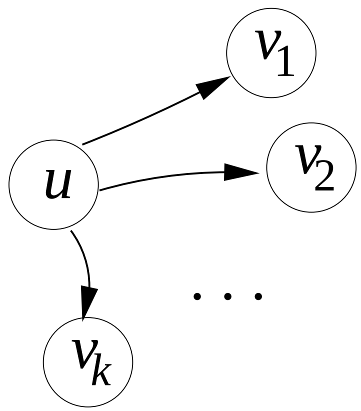
Base forwarding with retransmissions:

$$\boxed{u \to v^{a_v}}$$



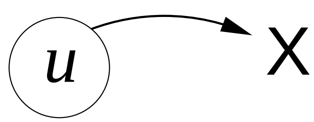
Sequential and parallel forwarding:

$$\boxed{u \to v_1^{a_1} v_2^{a_2} \cdots v_k^{a_k}}$$



Path completion:

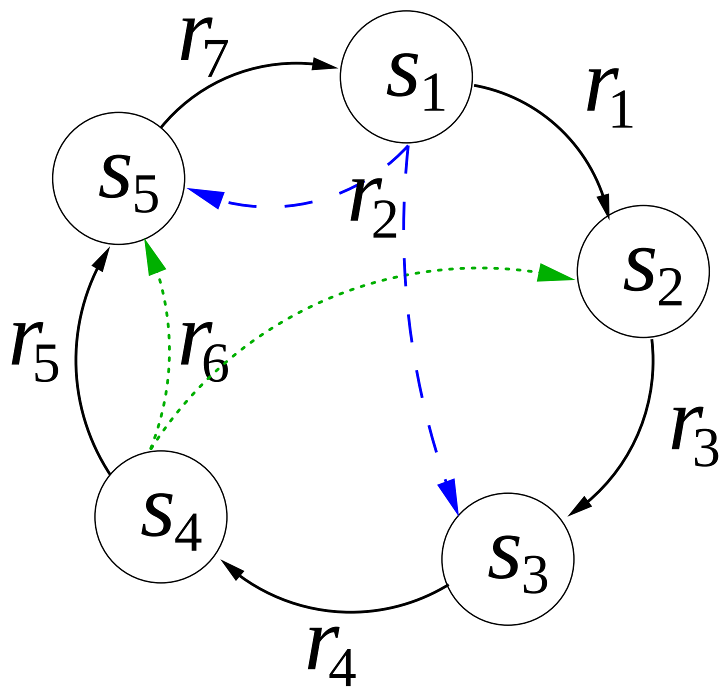
$$\boxed{u \to \varepsilon}$$



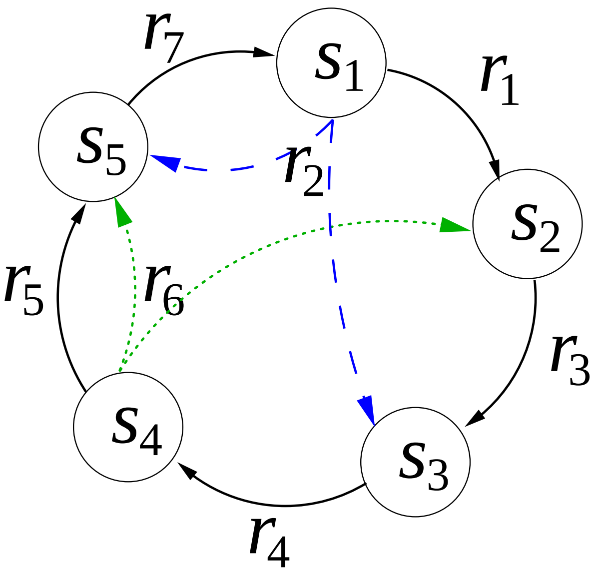
Routing Grammar: example

- Nodes $N = \{s_1, s_2, \ldots, s_5\}$
- Clockwise links (ring): r_1, r_3, r_4, r_5, r_7
- Parallel: r_2 (dashed, blue)
- Sequential: r_6 (dotted, green)
- σ_{seq} and σ_{par} mark sequential and parallel forwarding rules

$$
\begin{aligned}
r_1, r_2 :&\quad s_1 \rightarrow s_2 \mid \sigma_{\text{seq}} s_3 s_5 \\
r_3 :&\quad s_2 \rightarrow s_3 \\
r_4 :&\quad s_3 \rightarrow s_4 \\
r_5, r_6 :&\quad s_4 \rightarrow s_5 \mid \sigma_{\text{par}} s_2 s_5 \\
r_7 :&\quad s_5 \rightarrow s_1
\end{aligned}
$$

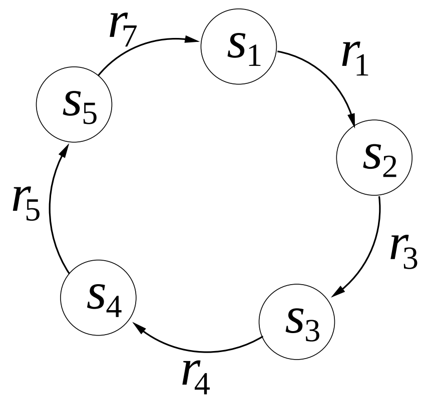


Routing Grammar: example, cont.



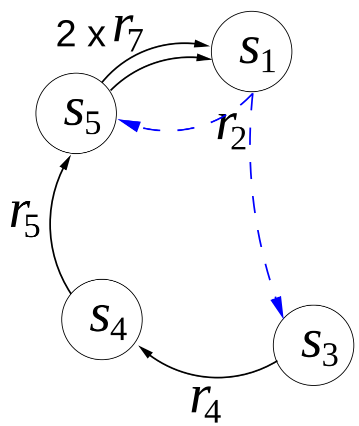
Cycles $(1,0,0,0,0) \Rightarrow^{+} (1,0,0,0,0)$

$$
\begin{aligned}
s_1 \overset{r_1}{\Rightarrow} s_2 \overset{r_3}{\Rightarrow} s_3 \overset{r_4}{\Rightarrow} \\
\Rightarrow s_4 \overset{r_5}{\Rightarrow} s_5 \overset{r_7}{\Rightarrow} s_1
\end{aligned}
$$



$$
\begin{aligned}
r_1, r_2 &: \quad s_1 \rightarrow s_2 \mid \sigma_{\text{seq}} s_3 s_5 \\
r_3 &: \quad s_2 \rightarrow s_3 \\
r_4 &: \quad s_3 \rightarrow s_4 \\
r_5, r_6 &: \quad s_4 \rightarrow s_5 \mid \sigma_{\text{par}} s_2 s_5 \\
r_7 &: \quad s_5 \rightarrow s_1 \\
r_8 &: \quad s_1 \rightarrow \varepsilon
\end{aligned}
$$

$$
s_1 \overset{r_2}{\Rightarrow} \sigma_{\text{seq}} \begin{array}{l} s_3 \overset{r_4}{\Rightarrow} s_4 \overset{r_5}{\Rightarrow} s_5 \overset{r_7}{\Rightarrow} s_1 \\ s_5 \overset{r_7}{\Rightarrow} s_1 \Rightarrow \varepsilon \end{array}
$$



Routing: Context-Free and Context-Dependent Forwarding Rules

Route $s_1^{b_1^-} \cdots s_l^{b_l^-} \Rightarrow^+ d_1^{b_1^+} \cdots d_k^{b_k^+}$

Advanced structure is due to hyper-arcs $(u; v_1, \ldots, v_k)$, in contrast to digraph arcs $u \to v$

Intermediaries u perform context-free forwarding rules $u \to v_1^{a_1} v_2^{a_2} \cdots v_k^{a_k}$

Generalization: $u_1^{a_1''} u_2^{a_2''} \cdots u_k^{a_l''} \to v_1^{a_1'} v_2^{a_2'} \cdots v_k^{a_k'}$

$$\sum_{r \in R} a_{ur} x_r = \sum_{r \in R_u} x_r, \quad u \in N \quad \rightsquigarrow \quad \sum_{r \in R} a'_{ur} x_r = \sum_{r \in R_u} a''_{ur} x_r, \quad u \in N$$

Or, since R_u and R_v may overlap for $u, v \in N$,

$$\sum_{j=1}^{m} a'_{ij} x_j = \sum_{j=1}^{m} a''_{ij} x_j, \quad i = 1, 2, \ldots, n, \tag{2}$$

Grammars and LDE Systems

$u_1^{a_1''} u_2^{a_2''} \cdots u_k^{a_l''} \to v_1^{a_1'} v_2^{a_2'} \cdots v_k^{a_k'}$

Close to Petri nets (equivalently, to Vector Addition Systems)

1. N is a set of *places*
2. R is a set of *transitions*
3. $W(u, r) : N \times R \to \mathbb{Z}_+$ is an input function (A'', a''_{ur})
4. $W(r, u) : N \times R \to \mathbb{Z}_+$ is an output function (A', a'_{ur})

$$\sum_{r \in R} a'_{ur} x_r = \sum_{r \in R_u} a''_{ur} x_r, \quad u \in N \tag{4}$$

Solutions to (4) are "invariants" of Petri net

Cyclic Structures in Routing

Route $s_1^{b_1^-} \cdots s_l^{b_l^-} \Rightarrow^+ d_1^{b_1^+} \cdots d_k^{b_k^+}$

If $s_i = d_j$ we have bidirectional communication

- Simple case: $s \Rightarrow^+ d \Rightarrow^+ s$ (one cycle covers multiple d)
- All-to-all workload: $(s)_{s \in N} \Rightarrow^+ (s)_{s \in N}$

In general:

$$(s^{b_u^-})_{s \in N} \Rightarrow^+ (s^{b_s^+})_{s \in N}$$

LDE system:

$$A'x + b^- = A''x + b^+$$

Conclusion

- General grammar-based case of LDE systems

$$\sum_{r \in R} a'_{ur} x_r = \sum_{r \in R_u} a''_{ur} x_r, \quad u \in N \tag{4}$$

- Practical algorithms (for Hilbert basis)
- Cyclic structures
  - particularization of A'' (e.g., sparse networks)
  - the role in routing
  - analysis (which elements of Hilbert basis)
  - construction as a composition of grammar sub-derivations (e.g., in CF-case: $u \Rightarrow^+ v$, $u \Rightarrow^+ u$, $u \Rightarrow^+ \varepsilon$)

Thank you!

Dmitry Korzun

dkorzun@cs.karelia.ru

Questions?