SORTING BY TRANSPOSITIONS IS DIFFICULT*

LAURENT BULTEAU, GUILLAUME FERTIN, IRENA RUSU†

Abstract. In comparative genomics, a transposition is an operation that exchanges two consecutive sequences of genes in a genome. The transposition distance between two genomes, that is, the minimum number of transpositions needed to transform a genome into another, is, according to numerous studies, a relevant evolutionary distance. The problem of computing this distance when genomes are represented by permutations is called the Sorting by Transpositions problem (SBT), and has been introduced by Bafna and Pevzner [3] in 1995. It has naturally been the focus of a number of studies, see for instance [17], but the computational complexity of this problem has remained undetermined for 15 years.

In this paper, we answer this long-standing open question by proving that the Sorting by Transpositions problem is NP-hard. As a corollary of our result, we also prove that the following problem, first described in [10], is NP-hard: given a permutation π, is it possible to sort π using exactly $d_b(\pi)/3$ transpositions, where $d_b(\pi)$ is the number of breakpoints of π?

Key words. Theoretical Aspects of Computational Biology, Comparative Genomics, Transposition Distance, Computational Complexity

AMS subject classifications. 05A05

Introduction. Along with reversals, transpositions are one of the most elementary large-scale operations that can affect a genome. A transposition consists in swapping two consecutive sequences of genes or, equivalently, in moving a sequence of genes from one place to another in the genome. The transposition distance between two genomes is the minimum number of such operations that are needed to transform one genome into the other. Computing this distance is a challenge in comparative genomics, see for instance [23], since it gives a maximum parsimony evolution scenario between the two studied genomes.

The Sorting by Transpositions problem is the problem of computing the transposition distance between genomes represented by permutations: see [17] for a detailed review on this problem and its variants. Since its introduction by Bafna and Pevzner [3, 4], the complexity of this problem has never been established. Hence a number of studies [4, 10, 18, 20, 14, 5, 16] aim at designing approximation algorithms or heuristics, the best known fixed-ratio algorithm being a 1.375-approximation [14]. Other works [19, 10, 15, 21, 14, 5] aim at computing bounds on the transposition distance of a permutation. Studies have also been devoted to variants of this problem, by considering, for example, sorting by prefix transpositions [13, 22, 8] (in which one of the transposed sequences has to be a prefix of the sequence), or sorting by transpositions in strings [11, 12, 26, 25] (where multiple occurrences of each element are allowed in the sequences), possibly using weighted or prefix transpositions [24, 6, 2, 8, 1]. Note also that sorting a permutation by block-interchanges (i.e., exchanges of not necessarily consecutive sequences) is solvable in polynomial time [9].

In this paper, we address the long-standing issue of determining the complexity class of the Sorting by Transpositions problem, by giving a polynomial-time reduction from SAT, thus proving the NP-hardness of this problem. Our reduction is

* A preliminary version of this paper appeared in the proceedings of the 38th International Colloquium on Automata, Languages and Programming, ICALP 2011 [7].

† Laboratoire d'Informatique de Nantes-Atlantique (LINA), UMR CNRS 6241 – Université de Nantes, 2 rue de la Houssinière, 44322 Nantes Cedex 3, France ({Laurent.Bulteau, Guillaume.Fertin, Irena.Rusu}@univ-nantes.fr).

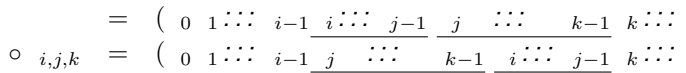


Fig. 1.1: Transposition $\tau_{i,j,k}$, with $0 < i < j < k \leq n$.

based on the study of transpositions that remove three breakpoints. A corollary of our result is the NP-hardness of the following problem, introduced in [10]: given a permutation π, is it possible to sort π using exactly $d_b(\pi)$=3 transpositions, where $d_b(\pi)$ is the number of breakpoints of π? The first section provides some preliminary definitions and results about the transposition distance and its relationship with breakpoints. In the second section, we introduce an intermediate structure used in our reduction, the 3DT-instance, and provide results giving some equivalences between 3DT-instances and permutations. The third section is the core of the reduction; it is devoted to the construction of "NP-hard" 3DT-instances, using an intricate assembling of basic blocks. Finally, the fourth section concludes the reduction by porting the previous construction to permutations, thus proving the NP-hardness of the Sorting By Transpositions problem.

1. Preliminaries.

1.1. Transpositions and Breakpoints. In this paper, n denotes a positive integer. Let $[\![a\,;b]\!] = \{x \in \mathbb{N} \mid a \leq x \leq b\}$, and Id_n be the identity permutation over $[\![0\,;n]\!]$. We consider only permutations of $[\![0\,;n]\!]$ such that 0 and n are fixed-points. Given a word $u_1\ u_2\ \cdots\ u_l$, a *subword* is a subsequence $u_{p_1}\ u_{p_2}\ \cdots\ u_{p_{l'}}$, where $1 \leq p_1 < p_2 < \cdots < p_{l'} \leq l$. A *factor* is a subsequence of contiguous elements, i.e., a subword with $p_{k+1} = p_k + 1$ for every $k \in [\![1\,;l'-1]\!]$.

A transposition is an operation that exchanges two consecutive factors of a sequence. As we only work with permutations, it is defined as a permutation $\tau_{i,j,k}$, which, once composed with a permutation π, realizes this operation (see Figure 1.1). The transposition $\tau_{i,j,k}$ is formally defined as follows.

Definition 1.1 (Transposition). *Given three integers $i;j;k$ such that $0 < i < j < k \leq n$, the* transposition $\tau_{i,j,k}$ *over $[\![0\,;n]\!]$ is the following permutation (we write $q(j) = k + i - j$):*

For any $0 \leq x < i$;	$\tau_{i,j,k}(x) = x$
For any $i \leq x < q(j)$;	$\tau_{i,j,k}(x) = x + j - i$
For any $q(j) \leq x < k$;	$\tau_{i,j,k}(x) = x + j - k$
For any $k \leq x \leq n$;	$\tau_{i,j,k}(x) = x$

Note that the inverse function of $\tau_{i,j,k}$ is also a transposition. More precisely, $\tau_{i,j,k}^{-1} = \tau_{i,q(j),k}$.

The following two properties directly follow from the definition of a transposition:

Property 1.2. *Let $\pi = \sigma\tau_{i,j,k}$ be a transposition, $q(j) = k+i-j$, and $u;v \in [\![0\,;n]\!]$*

be two integers such that $u < v$*. Then:*

$$\begin{aligned} (u) > (v) &\Leftrightarrow i \leq u < q(j) \leq v < k \\ {}^{-1}(u) > {}^{-1}(v) &\Leftrightarrow i \leq u < j \leq v < k \end{aligned}$$

PROPERTY 1.3. *Let* $\quad$ *be the transposition* $= {}_{i,j,k}$*, and write* $q(j) = k + i - j$*. For all* $x \in [\![1\,;\,n]\!]$*, the values of* $(x-1)$ *and* ${}^{-1}(x-1)$ *are the following:*

$$\begin{aligned} &\forall x \in \{i; q(j); k\}; \qquad (x-1) = (x) - 1 \\ &\forall x \in \{i; j; k\}; \qquad {}^{-1}(x-1) = {}^{-1}(x) - 1 \end{aligned}$$

$$\begin{aligned} (i-1) &= (q(j)) - 1 \qquad {}^{-1}(i-1) = {}^{-1}(j) - 1 \\ (q(j)-1) &= (k) - 1 \qquad {}^{-1}(j \end{aligned}$$

$$\begin{array}{rcl} & = & 0\ \underline{2\ 4}\ \underline{3\ 1}\ 5 \\ \circ\ _{1,3,5} & = & 0\ \underline{3}\ \underline{1\ 2}\ 4\ 5 \\ \circ\ _{1,3,5} \circ\ _{1,2,4} & = & 0\ 1\ 2\ 3\ 4\ 5 \end{array}$$

Fig. 1.2: The transposition distance from $\ = (0\ 2\ 4\ 3\ 1\ 5)$ to Id_5 is 2: it is at most 2 since $\ \circ\ _{1,3,5} \circ\ _{1,2,4} = Id_5$, and it cannot be less than 2 since Property 1.7 applies with $d_b(\ )=3 = 5=3 > 1$.

2. 3-Deletion and Transposition Operations. In this section, we introduce 3DT-instances, which are the cornerstone of our reduction from SAT to the Sorting by Transpositions problem, since they are used as an intermediate between instances of the two problems. We first define 3DT-instances and the possible operations that can be applied to them, then we focus on the equivalence between these instances and permutations. Section 2.4 stands out of the flow of the reduction, and gives a way to match 3DT-instances with the cycle graphs of 3-permutations. It should help the readers who are already familiar with the notion of cycle graphs following the definitions introduced here.

2.1. 3DT-instances. We introduce 3DT-instances as stand-alone mathematical objects, and give a set of tools to handle them. The two main objectives are (1) to accurately reflect the behavior of the breakpoints in a permutation such that $d_t(\ ) = d_b(\ )=3$ (this aspect will be formalized in the following sections) and (2) to make 3DT-instances easy to construct (they can be defined as a word whose letters can be partitioned into triples).

Definition 2.1 (3DT-instance). *A* 3DT-instance $I = \langle \Sigma; T; \ \rangle$ *of span* n *is composed of the following elements:*

- Σ*: an alphabet;*
- $T = \{(a_i; b_i; c_i) \mid 1 \leq i \leq |T|\}$*: a set of (ordered) triples of elements of* Σ*, partitioning* Σ *(i.e., all elements are pairwise distinct, and* $\bigcup_{i=1}^{|T|}\{a_i; b_i; c_i\} = \Sigma$*);*
- $\ : \Sigma \to [\![1\,;\, n]\!]$*, an injection.*

The domain *of* I *is the image of* $\ $*, that is the set* $L = \{\ (\ ) \mid \ \in \Sigma\}$.

The word representation *of* I *is the* n*-letter word* $u_1\ u_2 :: u_n$ *over* $\Sigma \cup \{\cdot\}$ *(where* $\cdot \notin \Sigma$*), such that for all* $i \in L$, $\ (u_i) = i$*, and for* $i \in [\![1\,;\, n]\!] - L$, $u_i = \cdot$.

Two examples of 3DT-instances are given in Example 2.1. Note that such instances can be defined by their word representation and by their set of triples T. The word representation of the empty 3DT-instance (in which $\Sigma = \emptyset$) is a sequence of n dot. We may also denote this instance by $"$.

Example 2.1. *In this example, we define two 3DT-instances of span 6,* $I = \langle \Sigma; T; \ \rangle$ *and* $I' = \langle \Sigma'; T'; \ '\rangle$*:*

$$\begin{array}{rcll} I & = & a_1\ c_2\ b_1\ b_2\ c_1\ a_2 & \text{with } T = \{(a_1; b_1; c_1); (a_2; b_2; c_2)\} \\ I' & = & \cdot\ b_2\ \cdot\ c_2\ \cdot\ a_2 & \text{with } T' = \{(a_2; b_2; c_2)\} \end{array}$$

Here, I *has an alphabet of size 6,* $\Sigma = \{a_1; b_1; c_1; a_2; b_2; c_2\}$*, hence* $\ $ *is a bijection (*$\ (a_1) = 1$, $\ (c_2) = 2$, $\ (b_1) = 3$*, etc). The second instance,* I'*, has an alphabet of size 3,* $\Sigma' = \{a_2; b_2; c_2\}$*, with* $\ '(b_2) = 2$, $\ '(c_2) = 4$, $\ '(a_2) = 6$.

PROPERTY 2.2. *Let* $I = \langle \Sigma; T; \ \rangle$ *be a 3DT-instance of span* n *with domain* L. *Then*

$$|\Sigma| = |L| = 3\,|T| \leq n.$$

Proof. We have $|\Sigma| = |L|$ since is an injection with image L. The triples of T partition Σ so $|\Sigma| = 3|T|$, and finally $L \subseteq [\![1\,;\, n]\!]$ so $|L| \leq n$. □

DEFINITION 2.3. *Let* $I = \langle \Sigma; T; \ \rangle$ *be a 3DT-instance. The injection gives a total order over* Σ, *written* $\prec_I$ *(or* $\prec$, *if there is no ambiguity), defined by*

$$\forall \ _1; \ _2 \in \Sigma; \quad \ _1 \prec_I \ _2 \Leftrightarrow \ (\ _1) < \ (\ _2) \tag{2.1}$$

Two elements $_1$ *and* $_2$ *of* Σ *are called* consecutive *if there exists no element* $x \in \Sigma$ *such that* $_1 \prec_I x \prec_I \ _2$. *In this case, we write* $_1 /_I \ _2$ *(or simply* $_1 / \ _2$*).*

An equivalent definition is that $_1 \prec \ _2$ if $_1 \ _2$ is a subword of the word representation of I. Also, $_1 / \ _2$ if the word representation of I contains a factor of the kind $_1 \cdot^* \ _2$ (where $\cdot^*$ represents any sequence of $l \geq 0$ dots).

Using the triples in T, we define a successor function over the domain L:

DEFINITION 2.4. *Let* $I = \langle \Sigma; T; \ \rangle$ *be a 3DT-instance with domain* L. *The function* $succ_I : L \to L$ *is defined by:*

$$\begin{aligned} \forall (a; b; c) \in T; \quad (a) &\mapsto \ (b) \\ (b) &\mapsto \ (c) \\ (c) &\mapsto \ (a) \end{aligned}$$

Function $succ_I$ is a bijection, with no fixed-points, and such that $succ_I \circ succ_I \circ succ_I$ is the identity over L. In Example 2.1, we have:

$$succ_I = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 3 & 6 & 5 & 2 & 1 & 4 \end{pmatrix} \text{ and } succ_{I'} = \begin{pmatrix} 2 & 4 & 6 \\ 4 & 6 & 2 \end{pmatrix}.$$

2.2. 3DT-steps. We define 3DT-steps on 3DT-instances in order to reflect transpositions removing three breakpoints on permutations. In both points of view, we apply a transposition for which the bounds need to verify certain conditions (being a well-ordered triple for 3DT-instances, or satisfying a correct pattern of values for permutations).

DEFINITION 2.5. *Let* $I = \langle \Sigma; T; \ \rangle$ *be a 3DT-instance, and* $(a; b; c)$ *be a triple of* T. *Write* $i = \min\{ \ (a); \ (b); \ (c)\}$, $j = succ_I(i)$, *and* $k = succ_I(j)$. *The triple* $(a; b; c) \in T$ *is* well-ordered *if we have* $i < j < k$. *In such a case, we write* $[a; b; c; \ ]$ *for the transposition* $_{i,j,k}$.

An equivalent definition is that $(a; b; c)$ is well-ordered iff one of abc, bca, cab is a subword of the word representation of I. In Example 2.1, $(a_1; b_1; c_1)$ is well-ordered in I: indeed, we have $i = \ (a_1)$, $j = \ (b_1)$, $k = \ (c_1)$, and $i < j < k$. The triple $(a_2; b_2; c_2)$ is also well-ordered in I' ($i = \ '(b_2) < j = \ '(c_2) < k = \ '(a_2)$), but not in I: $i = \ (c_2) < k = \ (b_2) < j = \ (a_2)$. In this example, we have $[a_1; b_1; c_1; \ ] = \ _{1,3,5}$ and $[a_2; b_2; c_2; \ '] = \ _{2,4,6}$.

DEFINITION 2.6 (3DT-step). *Let* $I = \langle \Sigma; T; \ \rangle$ *be a 3DT-instance with* $(a; b; c) \in T$ *a well-ordered triple. The* 3DT-step *of parameter* $(a; b; c)$ *is the operation written* $\xrightarrow{(a,\,b,\,c)}$, *transforming* I *into the 3DT-instance* $I' = \langle \Sigma'; T'; \ ' \rangle$ *such that:*

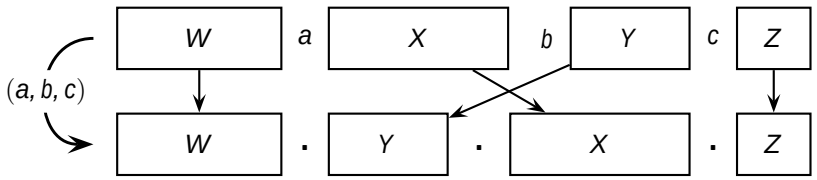


Fig. 2.1: The 3DT-step $\xrightarrow{(a,b,c)}$ has two effects, here represented on the word representation of a 3DT-instance: the triple $(a;b;c)$ is deleted (and replaced by dots in this word representation), and the factors X and Y are swapped.

- $\Sigma' = \Sigma - \{a;b;c\}$
- $T' = T - \{(a;b;c)\}$
- $\quad' : \begin{array}{ccc} \Sigma' & \to & [\![1\,;\,n]\!] \\ & \mapsto & {}^{-1}(\ \ (\ \ )) \end{array}$ $(with \quad = \quad [a;b;c;\ \ ]).$

A 3DT-step has two effects on a 3DT-instance, as represented in Figure 2.1. The first is to remove a necessarily well-ordered triple from T (hence from Σ). The second is, by applying a transposition to , to shift the position of some of the remaining elements. Note that a triple that is not well-ordered in I can become well-ordered in I', or vice-versa. In Example 2.1, I' can be obtained from I via a 3DT-step: $I \xrightarrow{(a_1,b_1,c_1)} I'$. Moreover, $I' \xrightarrow{(a_2,b_2,c_2)} "$. A more complex example is given in Figure 2.2.

Note that a 3DT-step transforms the function $succ_I$ into $succ_{I'} = {}^{-1} \circ succ_I \circ$, restricted to L', the domain of the new instance I'. Indeed, for all $(a;b;c) \in T'$, we have

$$\begin{aligned} succ_{I'}(\ \ '(a)) &= \ \ '(b) \\ &= {}^{-1}(\ \ (b)) \\ &= {}^{-1}(succ_I(\ \ (a))) \\ &= {}^{-1}(succ_I(\ \ (\ \ '(a)))) \\ &= (\ \ {}^{-1} \circ succ_I \circ \ \ )(\ \ '(a)) \end{aligned}$$

The computation is similar for $\ \ '(b)$ and $\ \ '(c)$.

DEFINITION 2.7 (3DT-collapsibility). *A 3DT-instance* $I = \langle \Sigma; T; \ \ \rangle$ *is* 3DT-collapsible *if there exists a sequence of 3DT-instances* $I_k; I_{k-1}; \dots; I_0$ *such that*

$$\begin{aligned} & I_k = I \\ & \forall i \in [\![1\,;\,k]\!]\,; \quad \exists (a;b;c) \in T; \quad I_i \xrightarrow{(a,b,c)} I_{i-1} \\ & I_0 = " \end{aligned}$$

In Example 2.1, I and I' are 3DT-collapsible, since $I \xrightarrow{(a_1,b_1,c_1)} I' \xrightarrow{(a_2,b_2,c_2)} "$. Another example is the 3DT-instance defined in Figure 2.2. Note that in the example of Figure 2.2, there are in fact two distinct paths leading to the empty instance.

2.3. Equivalence with the Transposition Distance. DEFINITION 2.8. *Let* $I = \langle \Sigma; T; \ \ \rangle$ *be a 3DT-instance of span* n *with domain* L*, and* *be a permutation of*

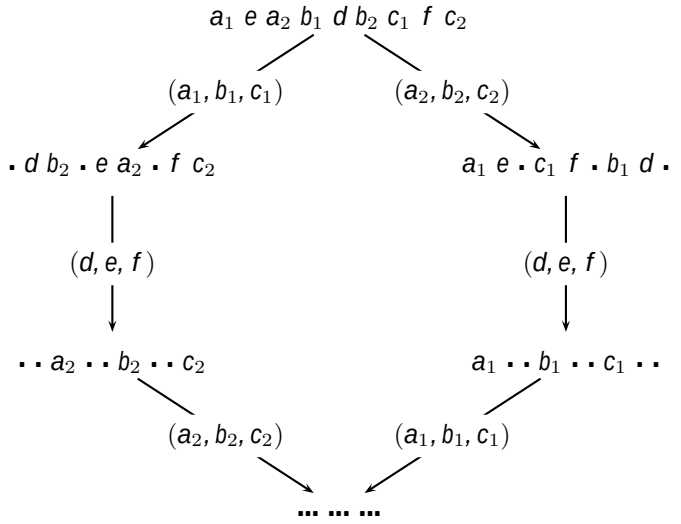


Fig. 2.2: Possible 3DT-steps from the instance I defined by the word $a_1\ e\ a_2\ b_1\ d\ b_2\ c_1\ f\ c_2$ and the set of triples $T = \{(a_1; b_1; c_1); (a_2; b_2; c_2); (d; e; f)\}$. We can see that there is a path from I to $''$, hence I is 3DT-collapsible. Note that both $(a_1; b_1; c_1)$ and $(a_2; b_2; c_2)$ are well-ordered in the initial instance, each one loses this property after applying the 3DT-step associated to the other, and becomes well-ordered again after applying the 3DT-step associated to $(d; e; f)$.

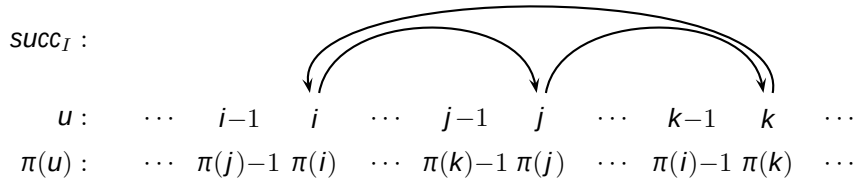


Fig. 2.3: Illustration of the equivalence $I \sim$ on three integers $(i; j; k)$ such that $j = succ_I(i)$ and $k = succ_I(j)$. It can be checked that $(v) = (u-1)+1$ for any $(u; v) \in \{(i; j); (j; k); (k; i)\}$.

$[\![0\,;\, n]\!]$. *We say that* I *and* *are equivalent, and we write* $I \sim$ *, if:*

$$\begin{array}{ll} & (0) = 0; \\ \forall v \in [\![1\,;\, n]\!] - L; & (v) = (v-1)+1; \\ \forall v \in L; & (v) = (succ_I^{-1}(v)-1)+1: \end{array}$$

With such an equivalence $I \sim$, the two following properties hold:

- The breakpoints of correspond to the elements of L (see Property 2.9).
- The triples of breakpoints that may be resolved immediately by a single transposition correspond to the well-ordered triples of T (see Figure 2.3 and Lemma 2.11).

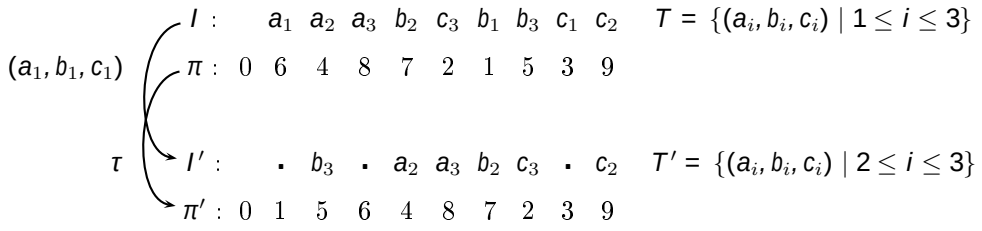


Fig. 2.4: Illustration of Lemma 2.10: since $I \sim$ and $I \xrightarrow{(a_1, b_1, c_1)} I'$, then $I' \sim \ ' = \ \circ \ $, where $\ = [a_1; b_1; c_1; \ ]$.

PROPERTY 2.9. *Let $I = \langle \Sigma; T; \ \rangle$ be a 3DT-instance of span n with domain L, and be a permutation of $[\![0\,;n]\!]$, such that $I \sim$. Then the number of breakpoints of is $d_b(\ ) = |L| = 3|T|$.*

Proof. Let $v \in [\![1\,;n]\!]$. By Definition 2.8, we have:

If $v \in L$, then $(v) = (v-1)+1$, so $(v-1; v)$ is an adjacency of .

If $v \in L$, we write $u = succ_I^{-1}(v)$, so $(v) = (u-1)+1$. Since $succ_I$ has no fixed-point, we have $u \neq v$, which implies $(u-1) \neq (v-1)$. Hence, $(v) \neq (v-1)+1$, and $(v-1; v)$ is a breakpoint of .

Consequently the number of breakpoints of is exactly $|L|$, and $|L| = 3|T|$ by Property 2.2. □

With the following lemma, we show that the equivalence between a 3DT-instance and a permutation is preserved after a 3DT-step, see Figure 2.4.

LEMMA 2.10. *Let $I = \langle \Sigma; T; \ \rangle$ be a 3DT-instance of span n, and be a permutation of $[\![0\,;n]\!]$, such that $I \sim$. If there exists a 3DT-step $I \xrightarrow{(a, b, c)} I'$ for some well-ordered triple $(a; b; c) \in T$, then I' and $' = \ \circ \ $, where $\ = [a; b; c; \ ]$, are equivalent.*

Proof. We write $(i; j; k)$ for the indices such that $\ = \ _{i,j,k}$ (by definition, $i = \min\{ \ (a); \ (b); \ (c)\}$, $j = succ_I(i)$, $k = succ_I(j)$). Since $(a; b; c)$ is well-ordered, we have $i < j < k$.

We have $I' = \langle \Sigma'; T'; \ ' \rangle$, with $\Sigma' = \Sigma - \{a; b; c\}$, $T' = T - \{(a; b; c)\}$, and $' : \ \mapsto \ ^{-1}(\ (\ ))$. We write respectively L and L' for the domains of I and I'. For all $v' \in [\![1\,;n]\!]$, we have

$$\begin{aligned} v' \in L' &\Leftrightarrow \exists \ \in \Sigma - \{a; b; c\};\ v' = \ ^{-1}(\ (\ )) \\ &\Leftrightarrow \ (v') \in L - \{i; j; k\} \end{aligned}$$

We prove the 3 required properties (see Definition 2.8) sequentially:

- $'(0) = \ (\ (0)) = \ (0) = 0$,
- $\forall v' \in [\![1\,;n]\!] - L'$, let $v = \ (v')$. Since $v' \notin L'$, we have either $v \in \{i; j; k\}$, or $v \notin L$. In the first case, we write $u = succ_I^{-1}(v)$ (then $u \in \{i; j; k\}$). By Property 1.3, $\ ^{-1}(u-1)$ is equal to $\ ^{-1}(succ_I(u)) - 1$, so $\ ^{-1}(u-1) = \ ^{-1}(v) - 1$. Hence,

$$\begin{aligned} '(v'-1)+1 &= \ (\ (\ ^{-1}(v) - 1)) + 1 \\ &= \ (u-1)+1 \\ &= \ (v) \text{ by Def. 2.8, since } v \in L \text{ and } v = succ_I(u) \\ &= \ '(v') \end{aligned}$$

In the second case, $v \in L$, we have

$$\begin{aligned}
\gamma'(v'-1)+1 &= (\ (\ \gamma^{-1}(v)-1))+1 \\
&= (\ (\ \gamma^{-1}(v-1)))+1 \text{ by Prop. 1.3, since } v \in \{i;j;k\} \\
&= (v-1)+1 \\
&= (v) \text{ by Def. 2.8, since } v \in L \\
&= \gamma'(v')
\end{aligned}$$

In both cases, we indeed have $\gamma'(v'-1)+1 = \gamma'(v')$.

- Let v' be an element of L'. We write $v = \gamma(v')$, $u = succ_I^{-1}(v)$, and $u' = \gamma^{-1}(u)$. Then $v' = \gamma^{-1}(succ_I(\gamma(u'))) = succ_{I'}(u')$. Moreover, $v \in \{i;j;k\}$, hence $u \in \{i;j;k\}$.

$$\begin{aligned}
\gamma'(u'-1)+1 &= (\ (\ \gamma^{-1}(u)-1))+1 \\
&= (\ (\ \gamma^{-1}(u-1)))+1 \text{ by Prop. 1.3, since } u \in \{i;j;k\} \\
&= (u-1)+1 \\
&= (v) \text{ by Def. 2.8, since } v \in L \text{ and } u = succ_I^{-1}(v) \\
&= (\ (\ \gamma^{-1}(v))) \\
&= \gamma'(v')
\end{aligned}$$

□

LEMMA 2.11. *Let $I = \langle \Sigma; T; \ \rangle$ be a 3DT-instance of span n, and $\ $ a permutation of $[\![0\,;n]\!]$, such that $I \sim \ $. If there exists a transposition $\ = \tau_{i,j,k}$ such that $d_b(\ \circ\ ) = d_b(\ ) - 3$, then T contains a well-ordered triple $(a;b;c)$ such that $\ = [a;b;c;\ ]$.*

Proof. We write $i' = \gamma^{-1}(i)$, $j' = \gamma^{-1}(j)$, and $k' = \gamma^{-1}(k)$. Note also that $i < j < k$.

Let $\gamma' = \gamma \circ \ $. For all $x \in [\![1\,;n]\!] - \{i;j;k\}$, we have, by Property 1.5, that $(x-1;x)$ is an adjacency of γ iff $(\ \gamma^{-1}(x)-1;\ \gamma^{-1}(x))$ is an adjacency of γ'. Hence, since $d_b(\ \gamma') = d_b(\ \gamma) - 3$, we necessarily have that $(i-1;i)$, $(j-1;j)$ and $(k-1;k)$ are breakpoints of γ, and $(i'-1;i')$, $(j'-1;j')$ and $(k'-1;k')$ are adjacencies of γ'. We have

$$\begin{aligned}
(i) &= (\ (i')) \\
&= \gamma'(i') \\
&= \gamma'(i'-1)+1 \text{ since } (i'-1;i') \text{ is an adjacency of } \gamma' \\
&= \gamma'(\ \gamma^{-1}(i)-1)+1 \\
&= \gamma'(\ \gamma^{-1}(k-1))+1 \text{ by Prop. 1.3} \\
&= (k-1)+1
\end{aligned}$$

Since $I \sim \ $ and $i \neq k$, by Definition 2.8, we necessarily have $i \in L$ (where L is the domain of I), and $i = succ_I(k)$.

Using the same method with $(j'-1;j')$ and $(k'-1;k')$, we obtain $j;k \in L$, $j = succ_I(i)$ and $k = succ_I(j)$. Hence, T contains one of the following three triples: $(\ \gamma^{-1}(i);\ \gamma^{-1}(j);\ \gamma^{-1}(k))$, $(\ \gamma^{-1}(j);\ \gamma^{-1}(k);\ \gamma^{-1}(i))$, or $(\ \gamma^{-1}(k);\ \gamma^{-1}(i);\ \gamma^{-1}(j))$. Writing $(a;b;c)$ this triple, we indeed have $\ \tau_{i,j,k} = [a;b;c;\ ]$ since $i < j < k$. □

THEOREM 2.12. *Let $I = \langle \Sigma; T; \ \rangle$ be a 3DT-instance of span n with domain L, and be a permutation of $[\![0\,;\,n]\!]$, such that $I \sim$. Then I is 3DT-collapsible if and only if $d_t(\ ) = |T| = d_b(\ )$=3.*

Proof. We prove the theorem by induction on $k = |T|$. For $k = 0$, necessarily $I = \varepsilon$ and $L = T = \emptyset$, and by Definition 2.8, $= Id_n$ ((0) = 0, and for all $v > 0$, $(v) =$ $(v-1)+1$). In this case, I is trivially 3DT-collapsible, and $d_t(\ ) = 0 = |T| = d_b(\ )$=3.

Suppose now $k = k' + 1$, with $k' \geq 0$, and the theorem is true for k'. By Property 2.9, we have $d_b(\ ) = 3k$, and by Property 1.7, $d_t(\ ) \geq 3k$=3 $= k$.

Assume first that I is 3DT-collapsible. Then there exist both a triple $(a; b; c) \in T$ and a 3DT-instance $I' = \langle \Sigma'; T'; \ ' \rangle$ such that $I \xrightarrow{(a,\,b,\,c)} I'$ and I' is 3DT-collapsible. Since $T' = T - \{(a; b; c)\}$, the size of T' is $k - 1 = k'$. By Lemma 2.10, we have $I' \sim \ ' = \ \circ$, with $= [a; b; c; \ ]$. Using the induction hypothesis, we know that $d_t(\ ') = k'$. So the transposition distance from $= \ ' \circ \ ^{-1}$ to the identity is at most, hence exactly, $k' + 1 = k$.

Assume now that $d_t(\ ) = k$. We can decompose into $= \ ' \circ \ ^{-1}$, where is a transposition and $'$ a permutation such that $d_t(\ ') = k - 1 = k'$. Since has $3k$ breakpoints (Property 2.9), and $' = \ \circ$ has at most $3k - 3$ breakpoints (Property 1.7), necessarily removes 3 breakpoints, and we can use Lemma 2.11: there exists a 3DT-step $I \xrightarrow{(a,\,b,\,c)} I'$, where $(a; b; c) \in T$ is a well-ordered triple and $= [a; b; c; \ ]$. We can now use Lemma 2.10, which yields $I' \sim \ ' = \ \circ$. Using the induction hypothesis, we obtain that I' is 3DT-collapsible, hence I is also 3DT-collapsible. This concludes the proof of the theorem. □

The previous theorem gives a way to reduce the problem of deciding if a 3DT-instance is collapsible to the SORTING BY TRANSPOSITIONS problem. However, it must be used carefully, since there exist 3DT-instances to which no permutation is equivalent (for example, $I = a_1\ a_2\ b_1\ b_2\ c_1\ c_2$ admits no permutation of $[\![0\,;\,6]\!]$ such that $I \sim$).

2.4. Parallel with the Cycle Graph. This section gives another point of view for the definition of 3DT-instances, in such a way that they (almost) match cycle graphs, as defined in [4]. It is given as a side remark and will not be used in the rest of the reduction, hence it is not described with an over-formal vocabulary, and we assume that the reader is already familiar with cycle graphs. This correspondence is summarized in Figure 2.5.

A 3DT-instance behaves similarly to a cycle graph containing only 1- and 3-cycles, that is, the cycle-graph of a so-called 3-permutation. Each $\in \Sigma$ corresponds to a reality arc in a 3-cycle (a triple corresponds to a 3-cycle), and each other position $i \in [\![1\,;\,n]\!]$ such that $u_i = \blacksquare$ corresponds to the reality arc of a 1-cycle. The $succ_I$ function corresponds to taking the next reality arc in a cycle. A triple is well-ordered if the corresponding cycle is oriented in the graph, and, for such triples, following a 3DT-step corresponds to breaking the 3-cycle into three 1-cycles (which may also change the orientation of some other cycles).

Given such a correspondence between a 3DT-instance I and the cycle graph G of a permutation , we obtain that I is equivalent to , and that I is 3DT-collapsible iff there exists a sequence of transpositions each breaking a 3-cycle of G (i.e., iff $d_t(\ ) = d_b(\ )$=3).

However, 3DT-instances are more general than cycle graphs: cycle graphs are defined from the permutations they represent while on the other hand 3DT-instances can be constructed without referring to a permutation, hence there is no cycle graph

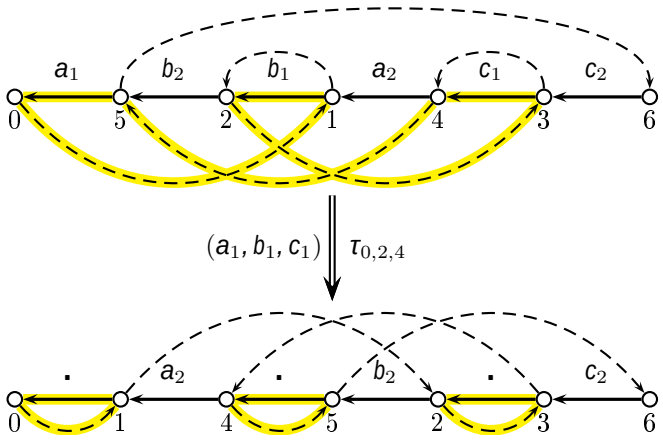


Fig. 2.5: The 3DT-instance $a_1\ b_2\ b_1\ a_2\ c_1\ c_2$ seen on the corresponding cycle graph, i.e., the cycle graph of permutation $(0;5;2;1;4;3;6)$. The ordered triple $(a_1;b_1;c_1)$ corresponds to the over-lined oriented 3-cycle, and following the 3DT-step removing this triple corresponds to breaking the cycle into 1-cycles.

corresponding to certain 3DT-instances (e.g., $I = a_1\ a_2\ b_1\ b_2\ c_1\ c_2$). Consequently, we can focus exclusively on the way triples are ordered and how they overlap – this task alone is the most complex one, see Section 3. The main drawback is that we need to verify, afterward, that there indeed exist permutations which are equivalent to the 3DT-instances we build (Section 4).

3. 3DT-collapsibility Is NP-Hard to Decide. In this section, we define, for any boolean formula , a corresponding 3DT-instance I_ϕ. We also prove that I_ϕ is 3DT-collapsible if and only if is satisfiable.

3.1. Block Structure. The construction of the 3DT-instance I_ϕ uses a decomposition into blocks, defined below. Some triples will be included in a block, in order to define its behavior, while others will be shared between two blocks, in order to pass information. The former are unconstrained, so that we can design blocks with the behavior we need (for example, blocks mimicking usual boolean functions), while the latter need to follow several rules, so that the blocks can conveniently be arranged together.

DEFINITION 3.1 (l-block decomposition). *An l-block decomposition $\mathcal{B}$ of a 3DT-instance I of span n is an l-tuple $(s_1;\dots;s_l)$ such that $s_1 = 0$, for all $h \in [\![1\,;\,l-1]\!]$, $s_h < s_{h+1}$ and $s_l < n$. We write $t_h = s_{h+1}$ for $h \in [\![1\,;\,l-1]\!]$, and $t_l = n$.*

Let $I = \langle \Sigma; T;\ \rangle$. For $h \in [\![1\,;\,l]\!]$, the factor $u_{s_h+1}\ u_{s_h+2}\ \dots u_{t_h}$ of the word representation $u_1\ u_2\ \dots u_n$ of I is called the full block $\mathcal{B}_h^{\bullet}$ *(it is a word over $\Sigma \cup \{\bullet\}$). The subword of $\mathcal{B}_h^{\bullet}$ where every occurrence of $\bullet$ is deleted is called the* block $\mathcal{B}_h$.

For $\in \Sigma$, we write $\mathit{block}_{I,\mathcal{B}}(\ ) = h$ if $(\ ) \in [\![s_h+1\,;\,t_h]\!]$ (equivalently, if appears in the word $\mathcal{B}_h$). A triple $(a;b;c) \in T$ is said to be internal *if $\mathit{block}_{I,\mathcal{B}}(a) = \mathit{block}_{I,\mathcal{B}}(b) = \mathit{block}_{I,\mathcal{B}}(c)$,* external *otherwise.*

If is a transposition such that for all $h \in [\![1\,;\,l]\!]$, $(s_h) <\ (t_h)$, we write $[\mathcal{B}]$ for the l-block decomposition $(\ (s_1);\dots;\ (s_l))$.

In the rest of this section, we mostly work with blocks instead of full blocks, since

we are only interested in the relative order of the elements, rather than their actual position. Full blocks are only used in definitions, where we want to control the dots in the word representation of the 3DT-instances we define. Note that, for $\alpha_1; \alpha_2 \in \Sigma$ such that $\mathit{block}_{I,\mathcal{B}}(\alpha_1) = \mathit{block}_{I,\mathcal{B}}(\alpha_2) = h$, the relation $\alpha_1 \,/\, \alpha_2$ is equivalent to $\alpha_1 \; \alpha_2$ *is a factor of* $\mathcal{B}_h$.

PROPERTY 3.2. *Let* $\mathcal{B} = (s_1; \dots; s_l)$ *be an* I*-block decomposition of a 3DT-instance of span* n*, and* $i; j; k \in [\![1\,;n]\!]$ *be three integers such that (a)* $i < j < k$ *and (b)* $\exists h_0$ *such that* $s_{h_0} < i < j \leq t_{h_0}$ *or* $s_{h_0} < j < k \leq t_{h_0}$ *(or both). Then for all* $h \in [\![1\,;l]\!]$, $\overset{-1}{i,j,k}(s_h) < \overset{-1}{i,j,k}(t_h)$*, and the* I*-block decomposition* $\overset{-1}{i,j,k}[\mathcal{B}]$ *is defined.*

Proof. For any $h \in [\![1\,;l]\!]$, we show that we cannot have $i \leq s_h < j \leq t_h < k$. Indeed, $s_h < j$ implies $h \leq h_0$ (since $s_h < j \leq t_{h_0} = s_{h_0+1}$), and $j \leq t_h$ implies $h \geq h_0$ (since $t_{h_0-1} = s_{h_0} < j \leq t_h$). Hence $s_h < j \leq t_h$ implies $h = h_0$, but $i \leq s_h; t_h < k$ contradicts both conditions $s_{h_0} < i$ and $k \leq t_{h_0}$: hence the relation $i \leq s_h < j \leq t_h < k$ is impossible.

By Property 1.2, since $s_h < t_h$ for all $h \in [\![1\,;l]\!]$, and $i \leq s_h < j \leq t_h < k$ does not hold, we have $\overset{-1}{i,j,k}(s_h) < \overset{-1}{i,j,k}(t_h)$, which is sufficient to define $\overset{-1}{i,j,k}[\mathcal{B}]$. □

The above property yields that, if $(a; b; c)$ is a well-ordered triple of a 3DT-instance $I = \langle \Sigma; T; \; \rangle$ ($=$ $[a; b; c;$]), and $\mathcal{B}$ is an I-block decomposition of I, then $^{-1}[\mathcal{B}]$ is defined if $(a; b; c)$ is an internal triple, or an external triple such that one of the following equalities is satisfied: $\mathit{block}_{I,\mathcal{B}}(a) = \mathit{block}_{I,\mathcal{B}}(b)$, $\mathit{block}_{I,\mathcal{B}}(b) = \mathit{block}_{I,\mathcal{B}}(c)$ or $\mathit{block}_{I,\mathcal{B}}(c) = \mathit{block}_{I,\mathcal{B}}(a)$. In this case, the 3DT-step $I \xrightarrow{(a,\,b,\,c)} I'$ is written $(I; \mathcal{B}) \xrightarrow{(a,\,b,\,c)} (I'; \mathcal{B}')$, where $\mathcal{B}' = \; ^{-1}[\mathcal{B}]$ is an I-block decomposition of I'.

DEFINITION 3.3 (Variable). *A* variable A *of a 3DT-instance* $I = \langle \Sigma; T; \; \rangle$ *is a pair of triples* $A = [(a; b; c); (x; y; z)]$ *of* T*. It is* valid *in an* I*-block decomposition* $\mathcal{B}$ *if*

(i) $\exists h_0 \in [\![1\,;l]\!]$ *such that* $\mathit{block}_{I,\mathcal{B}}(b) = \mathit{block}_{I,\mathcal{B}}(x) = \mathit{block}_{I,\mathcal{B}}(y) = h_0$

(ii) $\exists h_1 \in [\![1\,;l]\!]$, $h_1 \neq h_0$*, such that* $\mathit{block}_{I,\mathcal{B}}(a) = \mathit{block}_{I,\mathcal{B}}(c) = \mathit{block}_{I,\mathcal{B}}(z) = h_1$

(iii) *if* $x \prec y$*, then we have* $x \,/\, b \,/\, y$

(iv) $a \prec z \prec c$

For such a valid variable A*, the block* $\mathcal{B}_{h_0}$ *containing* $\{b; x; y\}$ *is called the* source *of* A *(we write* $\mathit{source}(A) = h_0$*), and the block* $\mathcal{B}_{h_1}$ *containing* $\{a; c; z\}$ *is called the* target *of* A *(we write* $\mathit{target}(A) = h_1$*). For* $h \in [\![1\,;l]\!]$*, the variables of which* $\mathcal{B}_h$ *is the source (resp. the target) are called the* output *(resp. the* input*) of* $\mathcal{B}_h$*. The 3DT-step* $I \xrightarrow{(x,\,y,\,z)} I'$ *is called the* activation *of* A *(it requires that* $(x; y; z)$ *is well-ordered).*

Note that since a valid variable $A = [(a; b; c); (x; y; z)]$ satisfies $\mathit{block}_{I,\mathcal{B}}(x) = \mathit{block}_{I,\mathcal{B}}(y)$, its activation can be written $(I; \mathcal{B}) \xrightarrow{(x,\,y,\,z)} (I'; \mathcal{B}')$.

EXAMPLE 3.1. *Consider the 3DT-instance* I *with the 2-block decomposition* $\mathcal{B}$ *such that:*

$$I = d\ y\ e\ x\ b\ f\ d'\ a\ f'\ z\ e'\ c$$
with triples $(a; b; c)$*;* $(d; e; f)$*;* $(d'; e'; f')$*;* $(x; y; z)$
$$\mathcal{B} = (0; 6)$$

This decomposition yields two blocks $\mathcal{B}_1 = d\,y\,e\,x\,b\,f$ *and* $\mathcal{B}_2 = d'\,a\,f'\,z\,e'\,c$*, which contain two internal triples* $(d; e; f)$ *and* $(d'; e'; f')$ *and two external triples forming a variable* $A = [(a; b; c); (x; y; z)]$*. We give below a sequence of 3DT-steps leading to the empty instance (for each step, the three deleted elements are in bold font, the elements that are swapped by the corresponding transposition are underlined, vertical bars give the limits of the blocks in the 2-block decomposition, and dot symbols are omitted). Note that in this example, variable* A *is valid, and remains valid until its activation.*

$$
\begin{aligned}
I &= |\, d\, \underline{y}\, e\, \underline{x}\, b\, \underline{\quad}\, f \,|\, d'\, a\, f'\, z\, e'\, c \,| \\
&\downarrow \quad (d, e, f) && \textit{Internal triple of } \mathcal{B}_1 \\
I_3 &= |\, x\, \underline{b}\, y \,|\, \underline{d'\, a\, f'}\, z\, e'\, c \,| \\
&\downarrow \quad (x, y, z) && \textit{Activation of } A \\
I_2 &= |\, "\,|\, d'\, a\, \underline{f'}\, b\, \underline{e'}\, c \,| \\
&\downarrow \quad (a, b, c) && \textit{Internal triple of } \mathcal{B}_2 \\
I_1 &= |\, "\,|\, d'\, e'\, f' \,| \\
&\downarrow \quad (d', e', f') && \textit{Internal triple of } \mathcal{B}_2 \\
I_0 &= |\, "\,|\, "\,| = "
\end{aligned}
$$

PROPERTY 3.4. *Let $(I; \mathcal{B})$ be a 3DT-instance with an I-block decomposition, and A be a variable of I that is valid in $\mathcal{B}$. Write $A = [(a; b; c); (x; y; z)]$. Then $(x; y; z)$ is well-ordered iff $x \prec y$; and $(a; b; c)$ is not well-ordered.*

Proof. Note that for all $\ ;\ ' \in \Sigma$, $\mathit{block}_{I,\mathcal{B}}(\ ) < \mathit{block}_{I,\mathcal{B}}(\ ') \Rightarrow \ \prec\ '$. Write $I = \langle \Sigma; T;\ \rangle$, $h_0 = \mathit{source}(A)$ and $h_1 = \mathit{target}(A)$: we have $h_0 \neq h_1$ by condition (ii) of Definition 3.3.

If $h_0 < h_1$, then, with condition (iv) of Definition 3.3, $b \prec a \prec c$, and either $x \prec y \prec z$ or $y \prec x \prec z$. Hence, $(a; b; c)$ is not well-ordered, and $(x; y; z)$ is well-ordered iff $x \prec y$.

Likewise, if $h_1 < h_0$, we have $a \prec c \prec b$, and $z \prec x \prec y$ or $z \prec y \prec x$. Again, $(a; b; c)$ is not well-ordered, and $(x; y; z)$ is well-ordered iff $x \prec y$. □

PROPERTY 3.5. *Let $(I; \mathcal{B})$ be a 3DT-instance with an I-block decomposition, such that the external triples of $I = \langle \Sigma; T;\ \rangle$ can be partitioned into a set of valid variables $\mathcal{A}$. Let $(d; e; f)$ be a well-ordered triple of I, such that there exists a 3DT-step $(I; \mathcal{B}) \xrightarrow{(d,\, e,\, f)} (I'; \mathcal{B}')$, with $I' = \langle \Sigma'; T';\ '\rangle$. Then one of the following two cases is true:*

- *$(d; e; f)$ is an internal triple. We write $h_0 = \mathit{block}_{I,\mathcal{B}}(d) = \mathit{block}_{I,\mathcal{B}}(e) = \mathit{block}_{I,\mathcal{B}}(f)$. Then for all $\ \in \Sigma'$, $\mathit{block}_{I',\mathcal{B}'}(\ ) = \mathit{block}_{I,\mathcal{B}}(\ )$. Moreover if $_1;\ _2 \in \Sigma'$ with $\mathit{block}_{I',\mathcal{B}'}(\ _1) = \mathit{block}_{I',\mathcal{B}'}(\ _2) \neq h_0$ and $\ _1 \prec_I \ _2$, then $\ _1 \prec_{I'} \ _2$.*
- *There exists a variable $A = [(a; b; c); (x; y; z)] \in \mathcal{A}$, such that $(d; e; f) = (x; y; z)$. Then $\mathit{block}_{I',\mathcal{B}'}(b) = \mathit{target}(A)$ and for all other $\ \in \Sigma' - \{b\}$, $\mathit{block}_{I',\mathcal{B}'}(\ ) = \mathit{block}_{I,\mathcal{B}}(\ )$. Moreover if $_1;\ _2 \in \Sigma' - \{b\}$, such that $\ _1 \prec_I \ _2$, then $\ _1 \prec_{I'} \ _2$.*

Proof. We write $\ $ for the transposition and $i; j; k$ for the three integers such that $\ = \ _{i,j,k} = \ [d; e; f;\ ]$ (necessarily, $0 < i < j < k \leq n$). We also write $\mathcal{B} = (s_0; s_1; \ldots; s_l)$. The triple $(d; e; f)$ is either internal or external in $\mathcal{B}$.

If $(d; e; f)$ is internal, with $h_0 = \mathit{block}_{I,\mathcal{B}}(d) = \mathit{block}_{I,\mathcal{B}}(e) = \mathit{block}_{I,\mathcal{B}}(f)$, we have (see Figure 3.1a):

$$s_{h_0} < i < j < k \leq t_{h_0}:$$

Hence for all $h \in [\![1\,;\, l]\!]$, either $s_h < i$ or $k \leq s_h$, and $\ ^{-1}(s_h) = s_h$ by Defini-

tion 1.1. Moreover, for all $\quad \in \Sigma$, we have

$$\begin{aligned} i \le \quad (\ ) < k \Rightarrow \quad & (\ ) \in [\![s_{h_0} + 1\,;\, t_{h_0}]\!] \\ & \text{and} \quad {}^{-1}(s_{h_0}) < i \le {}^{-1}(\ (\ )) < k \le {}^{-1}(t_{h_0}) \\ & \Rightarrow \mathit{block}_{I,\mathcal{B}}(\ ) = h_0 = \mathit{block}_{I',\mathcal{B}'}(\ ) \\ (\ ) < i \text{ or } k \le \quad (\ ) & \Rightarrow {}^{-1}(\ (\ )) = \quad (\ ) \\ & \Rightarrow \mathit{block}_{I',\mathcal{B}'}(\ ) = \mathit{block}_{I,\mathcal{B}}(\ ) \end{aligned}$$

Finally, if $\ _1;\ _2 \in \Sigma'$ with $\mathit{block}_{I',\mathcal{B}'}(\ _1) = \mathit{block}_{I',\mathcal{B}'}(\ _2) \neq h_0$, then we have both ${}^{-1}(\ (\ _1)) = \ (\ _1)$ and ${}^{-1}(\ (\ _2)) = \ (\ _2)$. Hence $\ _1 \prec_I \ _2 \Leftrightarrow \ _1 \prec_{I'} \ _2$.

If $(d;e;f)$ is external, then, since the set of external triples can be partitioned into variables, there exists a variable $A = [(a;b;c);(x;y;z)] \in \mathcal{A}$, such that $(d;e;f) = (a;b;c)$ or $(d;e;f) = (x;y;z)$. Since $(d;e;f)$ is well-ordered in I, we have, by Property 3.4, $(d;e;f) = (x;y;z)$ and $x \prec_I y$, see Figure 3.1b. And since A is valid, by condition (iv) of Definition 3.3, $x /_I b /_I y$. We write $h_0 = \mathit{source}(A)$ and $h_1 = \mathit{target}(A)$, and we assume that $h_0 < h_1$, which implies $x \prec_I y \prec_I z$ (the case $h_1 < h_0$ with $z \prec_I x \prec_I y$ is similar): thus, we have

$$i = \quad (x);\ j = \quad (y);\ k = \quad (z);\ \text{and } s_{h_0} < i < j \le t_{h_0} \le s_{h_1} < k \le t_{h_1}.$$

We define a set of indices U by

$$U = \{s_h \mid h \in [\![1\,;\, l]\!]\} \cup \{n\} \cup \{\ (\ ) \mid \quad \in \Sigma' - \{b\}\}.$$

We now show that for all $u \in U$, we have $u < i$ or $j \le u$. Indeed, if $u = s_h$ for some $h \in [\![1\,;\, l]\!]$, then either $h \le h_0$ and $u \le s_{h_0} < i$, or $h_0 < h$ and $j \le t_{h_0} \le u$. Also, if $u = n$, then $j \le u$. Finally, assume $u = \quad (\ )$, with $\quad \in \Sigma' - \{b\}$. We then have $x \prec_I \quad \prec_I y \Leftrightarrow \quad = b$, since $x /_I b /_I y$. Hence either $\quad \prec_I x$ and $u < \quad (x) = i$, or $y \prec_I \quad$ and $\quad (y) = j < u$.

By Property 1.2, if $u;v \in U$ are such that $u < v$, then ${}^{-1}(u) < {}^{-1}(v)$. This implies that elements of $\Sigma' - \{b\} = \Sigma - \{b;x;y;z\}$ do not change blocks after applying ${}^{-1}$ on $\quad$, and that the relative order of any two elements is preserved. Finally, for b, we have $x \prec_I b \prec_I y$, hence

$$i \le \quad (b) < j \le s_{h_1} < k \le t_{h_1}.$$

Thus, by Property 1.2, ${}^{-1}(s_{h_1}) < {}^{-1}(\ (b)) < {}^{-1}(t_{h_1})$, and $\mathit{block}_{I',\mathcal{B}'}(b) = h_1 = \mathit{target}(A)$. This completes the proof.□

Definition 3.6 (Valid context). *A 3DT-instance with an I-block decomposition $(I;\mathcal{B})$ is a* valid context *if the set of external triples of I can be partitioned into valid variables.*

With the following property, we ensure that a valid context remains *almost* valid after applying a 3DT-step: the partition of the external triples into variables is kept through this 3DT-step, but conditions (iii) and (iv) of Definition 3.3 are not necessarily satisfied for all variables.

Property 3.7. *Let $(I;\mathcal{B})$ be a valid context and $(I;\mathcal{B}) \xrightarrow{(d,e,f)} (I';\mathcal{B}')$ be a 3DT-step. Then the external triples of $(I';\mathcal{B}')$ can be partitioned into a set of variables, each satisfying conditions* (i) *and* (ii) *of Definition 3.3.*

Proof. Let $I = \langle \Sigma; T; \ \rangle$, $I' = \langle \Sigma'; T'; \ ' \rangle$, $\mathcal{A}$ be the set of variables of I, and E (resp. E') be the set of external triples of I (resp. I'). From Property 3.5, two cases are possible.

First case: $(d;e;f) \in E$. Then for all $\_ \in \Sigma'$, $block_{I',\mathcal{B}'}(\_) = block_{I,\mathcal{B}}(\_)$. Hence $E' = E$, and $(I';\mathcal{B}')$ has the same set of variables as $(I;\mathcal{B})$, that is $\mathcal{A}$. The source and target blocks of every variable remain unchanged, hence conditions (i) and (ii) of Definition 3.3 are still satisfied for each $A \in \mathcal{A}$ in $\mathcal{B}'$.

Second case: $(d;e;f) \in E$, and there exists a variable $A = [(a;b;c);(x;y;z)]$ in $\mathcal{A}$ such that $(d;e;f) = (x;y;z)$, by Property 3.5. Then $block_{I',\mathcal{B}'}(b) = target(A)$ and for all $\_ \in \Sigma' - \{b\}$, $block_{I',\mathcal{B}'}(\_) = block_{I,\mathcal{B}}(\_)$. Hence $block_{I',\mathcal{B}'}(b) = block_{I',\mathcal{B}'}(a) = block_{I',\mathcal{B}'}(c)$, and $E' = E - \{(x;y;z);(a;b;c)\}$: indeed, $(x;y;z)$ is deleted in T' so $(x;y;z) \in E'$, $(a;b;c)$ is internal in I', and every other triple is untouched. Finally, for every $A' = [(a';b';c');(x';y';z')] \in \mathcal{A} - \{A\}$, we have $block_{I',\mathcal{B}'}(\_) = block_{I,\mathcal{B}}(\_)$ for $\_ \in \{a';b';c';x';y';z'\}$, hence A' satisfies conditions (i) and (ii) of Definition 3.3 in $\mathcal{B}'$. □

Consider a block B in a valid context $(I;\mathcal{B})$ (there exists $h \in [\![1; l]\!]$ such that $B = \mathcal{B}_h$), and $(d;e;f)$ a triple of I such that $(I;\mathcal{B}) \xrightarrow{(d,e,f)} (I';\mathcal{B}')$ (we write $B' = \mathcal{B}'_h$). Then, following Property 3.5, four cases are possible:

- $h \in \{block_{I,\mathcal{B}}(d); block_{I,\mathcal{B}}(e); block_{I,\mathcal{B}}(f)\}$, hence $B' = B$, since, by Property 3.5, the relative order of the elements of B remains unchanged after the 3DT-step $\xrightarrow{(d,e,f)}$.
- $(d;e;f)$ is an internal triple of B. We write

$$\boxed{B} \text{ —— } (d,e,f) \longrightarrow \boxed{B'}$$

- $\exists A = [(a;b;c);(x;y;z)]$ such that $h = source(A)$ and $(d;e;f) = (x;y;z)$ (A is an output of B), see Figure 3.2 (left). We write

$$\boxed{B} \text{ ═══ } A \text{ ═══▶ } \boxed{B'}$$

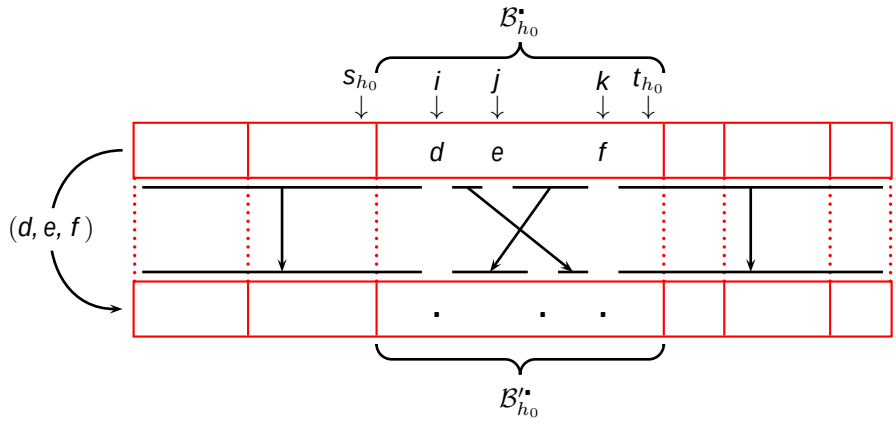
- $\exists A = [(a;b;c);(x;y;z)]$ such that $h = target(A)$ and $(d;e;f) = (x;y;z)$ (A is an input of B), see Figure 3.2 (right). We write

$$\boxed{B} \text{ ——— } A \longrightarrow \boxed{B'}$$

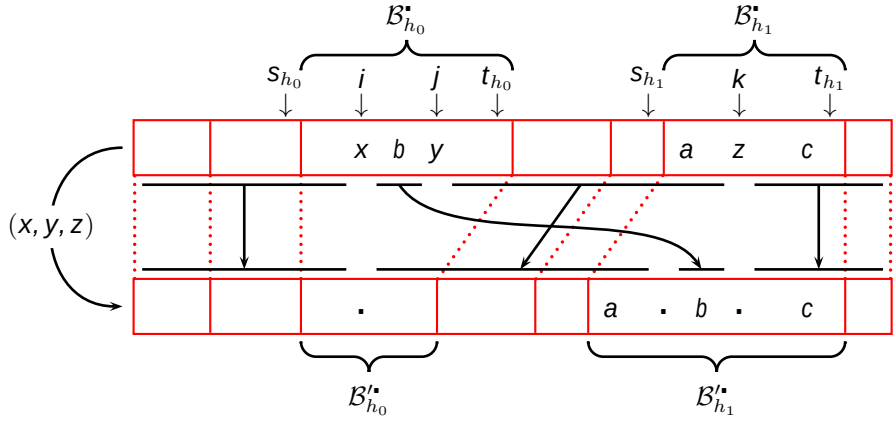
The graph obtained from a block B by following exhaustively the possible arcs as defined above (always assuming this block is in a valid context) is called the *behavior graph* of B.

3.2. Basic Blocks. We now define four basic blocks: copy, and, or, and var. They are studied independently in this section, before being assembled in Section 3.3. Each of these blocks is defined by a word and a set of triples. We distinguish internal triples, for which all three elements appear in a single block, from external triples, which are part of an input/output variable, and for which only one or two elements appear in the block. Note that each external triple is part of an input (resp. output) variable, which itself must be an output (resp. input) of another block, the other block containing the remaining elements of the triple.

We then draw the behavior graph of each of these blocks (Figures 3.3 to 3.6): in each case, we assume that the block is in a valid context, and follow exhaustively the 3DT-steps that can be applied to it. We then give another graph (Figures 3.7a to 3.7d), obtained from the behavior graph by contracting all arcs corresponding to 3DT-steps using internal triples, i.e., we assimilate every pair of nodes linked by such an arc. Hence, only the arcs corresponding to the activation of an input/output variable remain. From this second figure, we derive a property describing the behavior of the block, in terms of activating input and output variables (always provided this



(a)



(b)

Fig. 3.1: Effects of a 3DT-step $\xrightarrow{(d,e,f)}$ on an I-block decomposition if (a) $(d;e;f)$ is an internal triple, or (b) there exists a variable $A = [(a;b;c);(x;y;z)]$ such that $(d;e;f) = (x;y;z)$. Both figures are in fact derived from Figure 2.1 in the context of an I-block decomposition.

block is in a valid context). It must be kept in mind that for any variable, it is the state of the source block which determines whether it can be activated, whereas the activation itself affects mostly the target block.

3.2.1. The Block copy. This block aims at duplicating a variable: any of the two output variables can only be activated after the input variable has been activated.
Input variable: $A = [(a;b;c);(x;y;z)]$.
Output variables: $A_1 = [(a_1;b_1;c_1);(x_1;y_1;z_1)]$ and $A_2 = [(a_2;b_2;c_2);(x_2;y_2;z_2)]$.
Internal triple: $(d;e;f)$.
Definition:

$$[A_1;A_2] = \mathsf{copy}(A) \quad = \quad a\ y_1\ e\ z\ d\ y_2\ x_1\ b_1\ c\ x_2\ b_2\ f$$

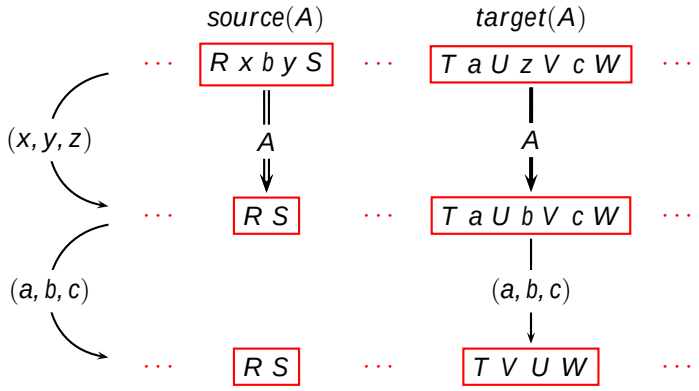


Fig. 3.2: The activation of a variable $A = [(a;b;c);(x;y;z)]$ is written with a double arc in the behavior graph of the source block of A and with a thick arc in the behavior graph of its target block. It can be followed by the 3DT-step $\xrightarrow{(a,b,c)}$, impacting only the target block of A. Dot symbols ($\cdot$) are omitted. We denote by $R;S;T;U;V;W$ some factors of the source and target blocks of A: the consequence of activating A is to allow U and V to be swapped in $\mathit{target}(A)$.

PROPERTY 3.8. *In a block* $[A_1;A_2] = \mathsf{copy}(A)$ *in a valid context, the possible orders in which* A, A_1 *and* A_2 *can be activated are* $(A;A_1;A_2)$ *and* $(A;A_2;A_1)$.

Proof. See Figures 3.3 and 3.7a. □

3.2.2. The Block and**.** This block aims at simulating a conjunction: the output variable can only be activated after both input variables have been activated.
Input variables: $A_1 = [(a_1;b_1;c_1);(x_1;y_1;z_1)]$ and $A_2 = [(a_2;b_2;c_2);(x_2;y_2;z_2)]$.
Output variable: $A = [(a;b;c);(x;y;z)]$.
Internal triple: $(d;e;f)$.
Definition:

$$A = \mathsf{and}(A_1;A_2) \quad = \quad a_1\ e\ z_1\ a_2\ c_1\ z_2\ d\ y\ c_2\ x\ b\ f$$

PROPERTY 3.9. *In a block* $A = \mathsf{and}(A_1;A_2)$ *in a valid context, the possible orders in which* A, A_1 *and* A_2 *can be activated are* $(A_1;A_2;A)$ *and* $(A_2;A_1;A)$.

Proof. See Figures 3.4 and 3.7b. □

3.2.3. The Block or**.** This block aims at simulating a disjunction: the output variable can be activated as soon as any of the two input variables is activated.
Input variables: $A_1 = [(a_1;b_1;c_1);(x_1;y_1;z_1)]$ and $A_2 = [(a_2;b_2;c_2);(x_2;y_2;z_2)]$.
Output variable: $A = [(a;b;c);(x;y;z)]$.
Internal triples: $(a';b';c')$ and $(d;e;f)$.
Definition:

$$A = \mathsf{or}(A_1;A_2) \quad = \quad a_1\ b'\ z_1\ a_2\ d\ y\ a'\ x\ b\ f\ z_2\ c_1\ e\ c'\ c_2$$

PROPERTY 3.10. *In a block* $A = \mathsf{or}(A_1;A_2)$ *in a valid context, the possible orders in which* A, A_1 *and* A_2 *can be activated are* $(A_1;A;A_2)$, $(A_2;A;A_1)$, $(A_1;A_2;A)$ *and* $(A_2;A_1;A)$.

Proof. See Figures 3.5 and 3.7c. □

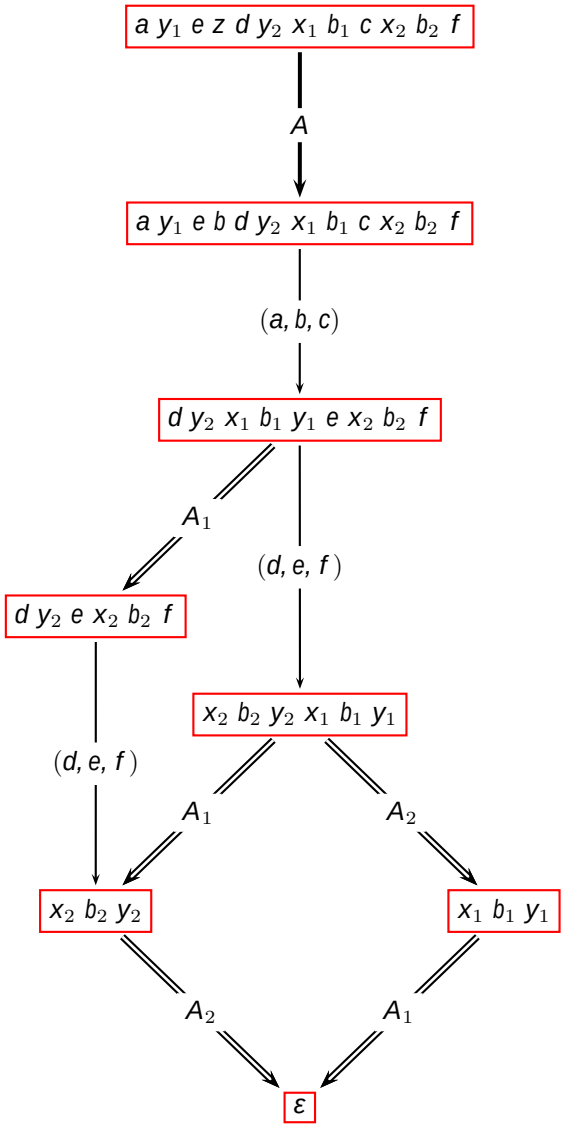


Fig. 3.3: Behavior graph of the block $[A_1; A_2] = \mathsf{copy}(A)$. A thick (resp. double) arc corresponds to the 3DT-step $\xrightarrow{(x,\,y,\,z)}$ for an input (resp. output) variable $[(a;b;c);(x;y;z)]$.

3.2.4. The Block var**.** This block aims at simulating a boolean variable: in a first stage, only one of the two output variables can be activated. The other needs the activation of the input variable to be activated.

Input variable: $A = [(a;b;c);(x;y;z)]$.

Output variables: $A_1 = [(a_1;b_1;c_1);(x_1;y_1;z_1)]$, $A_2 = [(a_2;b_2;c_2);(x_2;y_2;z_2)]$.

Internal triples: $(d_1;e_1;f_1)$, $(d_2;e_2;f_2)$, and $(a';b';c')$.

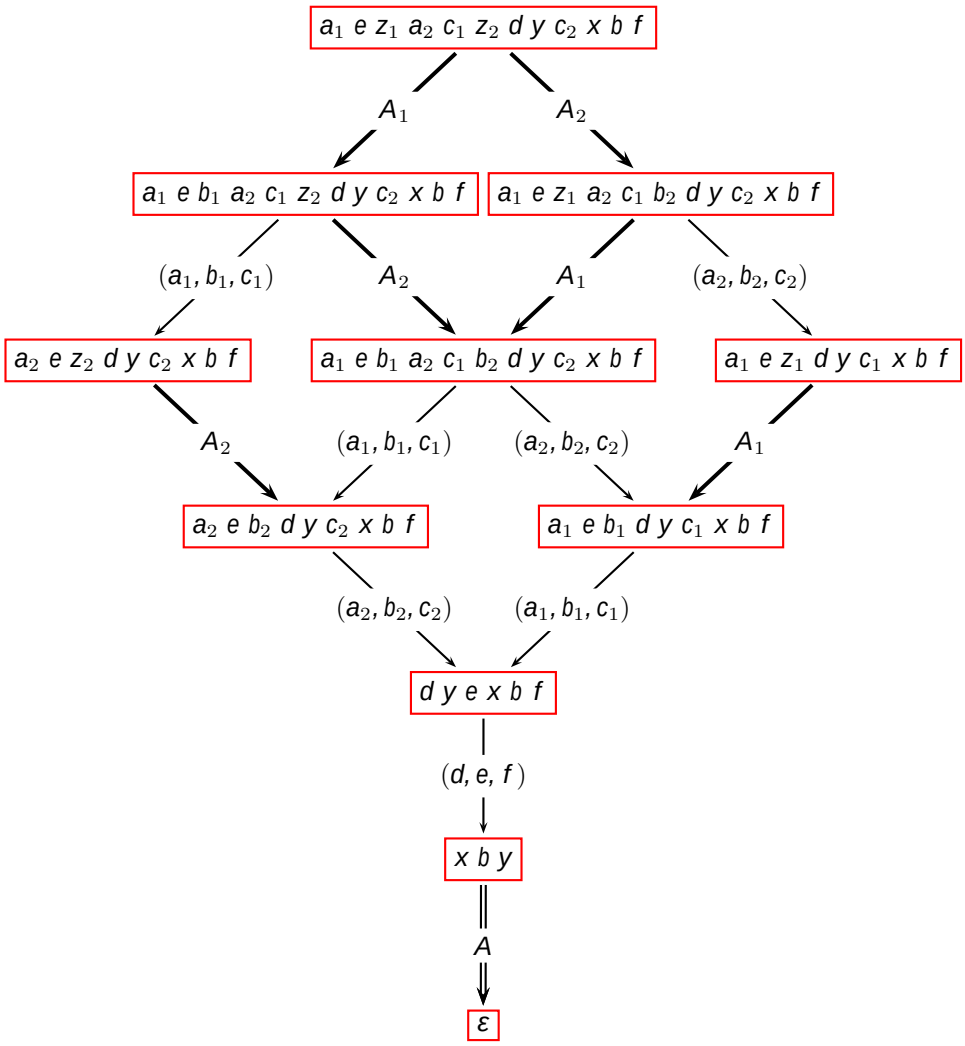


Fig. 3.4: Behavior graph of the block $A = \mathsf{and}(A_1; A_2)$.

Definition:

$$[A_1; A_2] = \mathsf{var}(A) \quad = \quad d_1\ y_1\ a\ d_2\ y_2\ e_1\ a'\ e_2\ x_1\ b_1\ f_1\ c'\ z\ b'\ c\ x_2\ b_2\ f_2$$

PROPERTY 3.11. *In a block $[A_1; A_2] = \mathsf{var}(A)$ in a valid context, the possible orders in which A, A_1 and A_2 can be activated are $(A_1; A; A_2)$, $(A_2; A; A_1)$, $(A; A_1; A_2)$ and $(A; A_2; A_1)$.*

Proof. See Figures 3.6 and 3.7d. □ With such a block, if A is not activated first, one needs to make a choice between activating A_1 or A_2. Once A is activated, however, all the remaining output variables are activable.

3.2.5. Assembling the Blocks copy, and, or, var.. DEFINITION 3.12 (Assembling of basic blocks). *An* assembling of basic blocks $(I; \mathcal{B})$ *is composed of a 3DT-instance I and an I-block decomposition $\mathcal{B}$ obtained by the following process:*

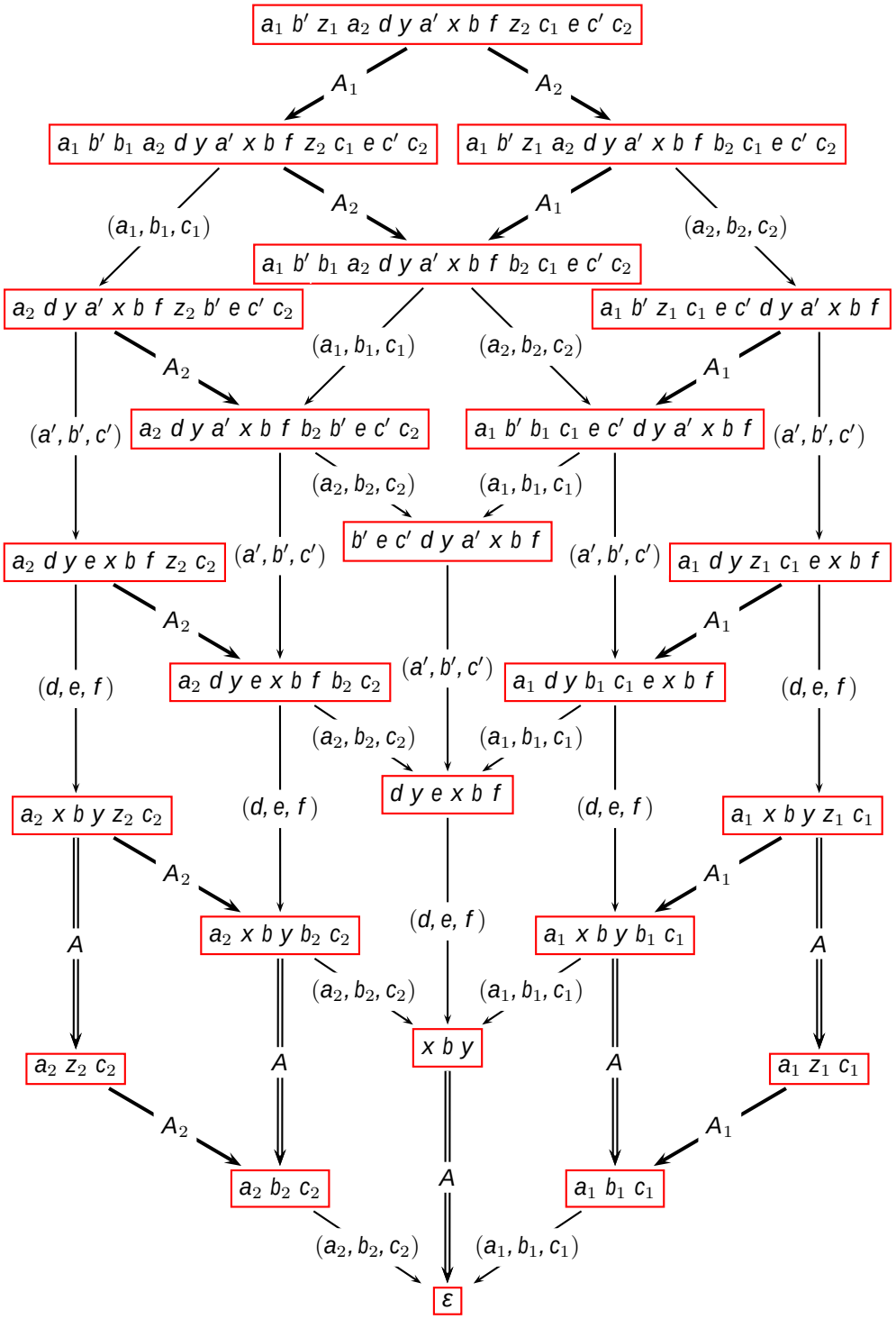


Fig. 3.5: Behavior graph of the block $A = \mathsf{or}(A_1; A_2)$.

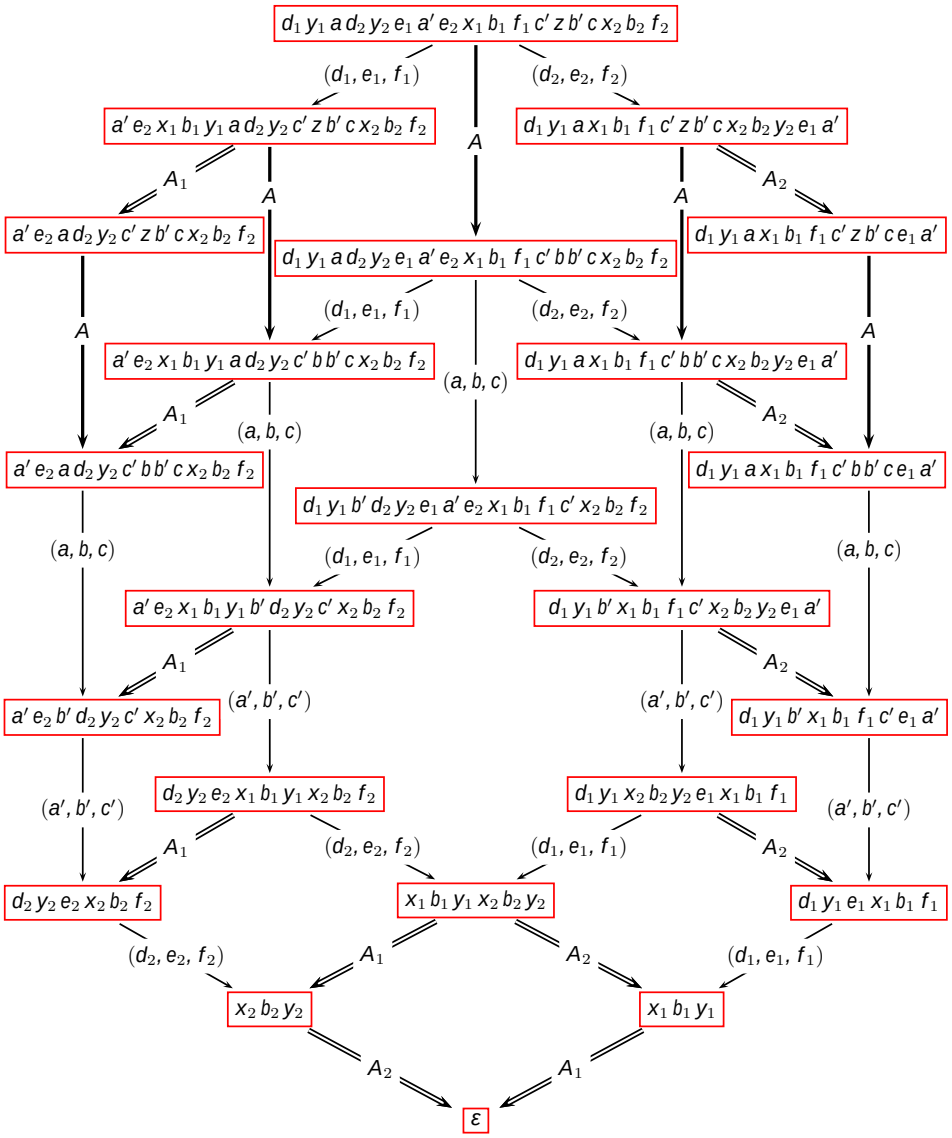
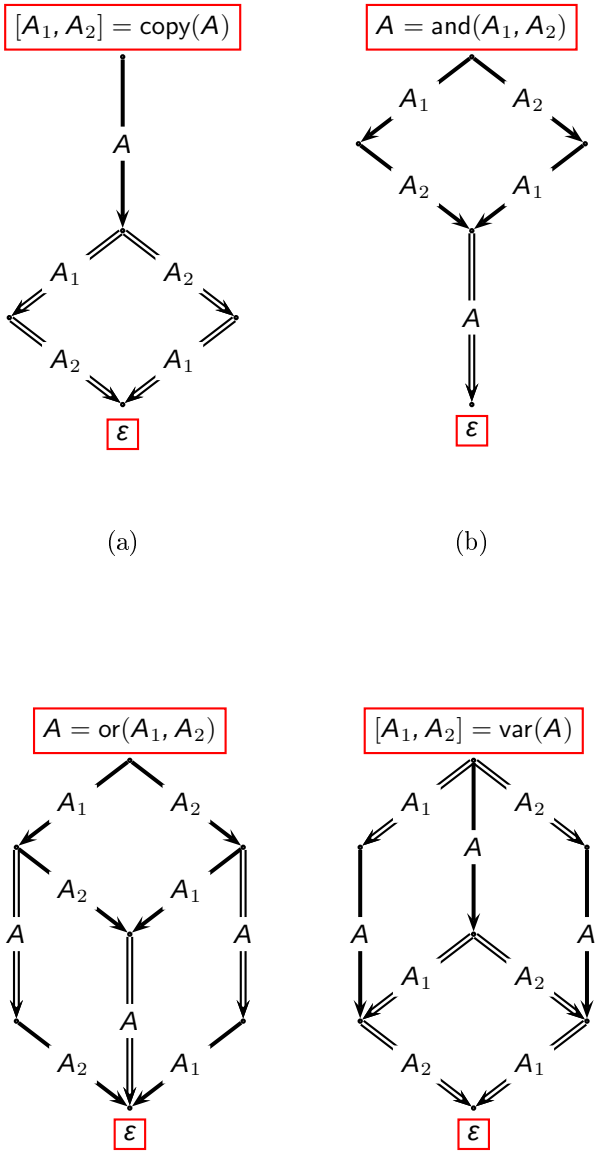


Fig. 3.6: Behavior graph of the block $[A_1; A_2] = \mathsf{var}(A)$.

- *Create a set of variables* $\mathcal{A}$.
- *Define* $I = \langle \Sigma; T; \ \rangle$ *by its word representation, as a concatenation of* l *factors* $\mathcal{B}_1^{\blacksquare}\ \mathcal{B}_2^{\blacksquare}\ ::: \ \mathcal{B}_l^{\blacksquare}$ *and a set of triples* T*, where each* $\mathcal{B}_h^{\blacksquare}$ *is one of the blocks* $[A_1; A_2] = \mathsf{copy}(A)$*,* $A = \mathsf{and}(A_1; A_2)$*,* $A = \mathsf{or}(A_1; A_2)$ *or* $[A_1; A_2] = \mathsf{var}(A)$*, with* $A_1; A_2; A \in \mathcal{A}$ *(such that each* $X \in \mathcal{A}$ *appears in the input of exactly one block, and in the output of exactly one other block); and where* T *is the union of the set of internal triples needed in each block, and the set of external triples defined by the variables of* $\mathcal{A}$.

EXAMPLE 3.2. *We create a 3DT-instance* I *with a 2-block decomposition* $\mathcal{B}$ *such*



(a) (b)

(c) (d)

Fig. 3.7: Abstract representations of the blocks copy, and, or, and var, obtained from each behavior graph (Figures 3.3, 3.4, 3.5, and 3.6) by contracting arcs corresponding to internal triples, and keeping only the arcs corresponding to variables. We see, for each block, which output variables are activable, depending on which variables have been activated.

that $(I; \mathcal{B})$ *is an assembling of basic blocks, defined as follows:*

- I *uses three variables,* $\mathcal{A} = \{X_1; X_2; Y\}$
- *the word representation of* I *is the concatenation of* $[X_1; X_2] = \mathsf{var}(Y)$ *and* $Y = \mathsf{or}(X_1; X_2)$

With variables $X_1 = [(a_1; b_1; c_1); (x_1; y_1; z_1)]$, $X_2 = [(a_2; b_2; c_2); (x_2; y_2; z_2)]$, $Y = [(a; b; c); (x; y; z)]$, *and the internal triples* $(d_1; e_1; f_1); (d_2; e_2; f_2); (a'; b'; c')$ *for the block* var, *and* $(a''; b''; c''); (d; e; f)$ *for the block* or, *the word representation of* I *is the following (note that its 2-block decomposition, emphasized with the vertical bars, is* $(0; 18)$*):*

$$I = |d_1\ y_1\ a\ d_2\ y_2\ e_1\ a'\ e_2\ x_1\ b_1\ f_1\ c'\ z\ b'\ c\ x_2\ b_2\ f_2|a_1\ b''\ z_1\ a_2\ d\ y\ a''\ x\ b\ f\ z_2\ c_1\ e\ c''\ c_2|$$

A possible sequence of 3DT-steps leading from I *to* $''$ *is given in Figure 3.8, hence* I *is 3DT-collapsible.*

LEMMA 3.13. *Let* I' *be a 3DT-instance with an* I*-block decomposition* $\mathcal{B}'$*, such that* $(I'; \mathcal{B}')$ *is obtained from an assembling of basic blocks* $(I; \mathcal{B})$ *after any number of 3DT-steps, i.e., there exist* $k \geq 0$ *triples* $(d_i; e_i; f_i)$, $i \in [\![1; k]\!]$, *such that* $(I; \mathcal{B}) \xrightarrow{(d_1, e_1, f_1)} \cdots \xrightarrow{(d_k, e_k, f_k)} (I'; \mathcal{B}')$.

Then $(I'; \mathcal{B}')$ *is a valid context. Moreover, if the set of variables of* $(I'; \mathcal{B}')$ *is empty, then* I' *is 3DT-collapsible.*

Proof. Write $\mathcal{A}$ the set of variables used to define $(I; \mathcal{B})$. We write $I = \langle \Sigma; T;\ \rangle$ and $I' = \langle \Sigma'; T';\ '\rangle$. We prove that $(I'; \mathcal{B}')$ is a valid context by induction on k (the number of 3DT-steps between $(I; \mathcal{B})$ and $(I'; \mathcal{B}')$). We also prove that for each $h \in [\![1; l]\!]$, $\mathcal{B}'_h$ appears as a node in the behavior graph of $\mathcal{B}_h$.

Suppose first that $k = 0$. We show that the set of external triples of $(I; \mathcal{B}) = (I'; \mathcal{B}')$ can be partitioned into valid variables, namely into $\mathcal{A}$. Indeed, from the definition of each block, for each $\in \Sigma$, is either part of an internal triple, or appears in a variable $A \in \mathcal{A}$. Conversely, for each $A = [(a; b; c); (x; y; z)] \in \mathcal{A}$, b, x and y appear in the block having A for output, and a, c and z appear in the block having A for input. Hence $(a; b; c)$ and $(x; y; z)$ are indeed two external triples of $(I; \mathcal{B})$. Hence each variable satisfies conditions (i) and (ii) of Definition 3.3. Conditions (iii) and (iv) can be checked in the definition of each block: we have, for each output variable, $y \prec x$, and for each input variable, $a \prec z \prec c$. Finally, each $\mathcal{B}_h$ appears in its own behavior graph.

Suppose now that $(I'; \mathcal{B}')$ is obtained from $(I; \mathcal{B})$ after k 3DT-steps, $k > 0$. Then there exists a 3DT-instance with an I-block decomposition $(I''; \mathcal{B}'')$ such that:

$$(I; \mathcal{B}) \xrightarrow{(d_1, e_1, f_1)} \cdots \xrightarrow{(d_{k-1}, e_{k-1}, f_{k-1})} (I''; \mathcal{B}'') \xrightarrow{(d_k, e_k, f_k)} (I'; \mathcal{B}'):$$

Consider $h \in [\![1; l]\!]$. By induction hypothesis, since $\mathcal{B}''_h$ is in a valid context $(I''; \mathcal{B}'')$, then, depending on $(d_k; e_k; f_k)$, either $\mathcal{B}'_h = \mathcal{B}''_h$, or there is an arc from $\mathcal{B}''_h$ to $\mathcal{B}'_h$ in the behavior graph. Hence $\mathcal{B}'_h$ is indeed a node in this graph. By Property 3.7, we know that the set of external triples of $(I'; \mathcal{B}')$ can be partitioned into variables satisfying conditions (i) and (ii) of Definition 3.3. Hence we need to prove that each variable satisfies conditions (iii) and (iv): by inspecting each node of the behavior graph, we verify that $x \prec y \Rightarrow x \,/\, b \,/\, y$ (resp. $a \prec z \prec c$) for each output (resp. input) variable $A = [(a; b; c); (x; y; z)]$ of the block. This concludes the induction proof.

We finally need to consider the case where the set of variables of $(I'; \mathcal{B}')$ is empty. Then for each $h \in [\![1; l]\!]$ we either have $\mathcal{B}'_h = ''$, or $\mathcal{B}'_h = a_h\ b_h\ c_h$ for some internal triple $(a_h; b_h; c_h)$ (in the case where $\mathcal{B}_h$ is a block or). Then $(I'; \mathcal{B}')$ is indeed 3DT-collapsible: simply follow in any order the 3DT-step $\xrightarrow{(a_h, b_h, c_h)}$ for each remaining triple $(a_h; b_h; c_h)$. □

$I = | \, \boldsymbol{d_1} \underline{y_1 \, a \, d_2 \, y_2} \, e$

$x_1, \ldots, x_m$, we always use the term *boolean variable* for the latter.

The 3DT-instance I_ϕ is defined as an assembling of basic blocks: we first define a set of variables, then we list the blocks of which the word representation of I_ϕ is the concatenation. It is necessary that each variable is part of the input (resp. the output) of exactly one block. Note that the relative order of the blocks is of no importance. We simply try, for readability reasons, to ensure that the source of a variable appears before its target, whenever possible. We say that a variable *represents* a term, i.e., a literal, clause or formula, if it can be activated if and only if this term is true (for some fixed assignment of the boolean variables), or if is satisfied by this assignment. We also say that a block *defines* a variable if it is the source block of this variable.

The construction of I_ϕ is done as follows (see Figure 3.9 for an example):

- Create a set of variables:
  - For each $i \in [\![1\,;\,m]\!]$, create $q_i + 1$ variables representing x_i: X_i and X_i^j, $j \in [\![1\,;\,q_i]\!]$, and $\bar{q}_i + 1$ variables representing $\neg x_i$: $\bar{X}_i$ and $\bar{X}_i^j$, $j \in [\![1\,;\,\bar{q}_i]\!]$.
  - For each $c \in [\![1\,;\quad]\!]$, create a variable Γ_c representing the clause C_c.
  - Create $m+1$ variables, A_ϕ and A_ϕ^i, $i \in [\![1\,;\,m]\!]$, representing the formula . We will show that A_ϕ has a key role in the construction: it can be activated only if is satisfiable, and, once activated, it allows every remaining variable to be activated.
  - We also use a number of intermediate variables, with names U_i^j, $\bar{U}_i^j$, V_c^i, W_c, and Y_i.
- Start with an empty 3DT-instance $''$, and add blocks successively:
  - For each $i \in [\![1\,;\,m]\!]$, add the following $q_i + \bar{q}_i - 1$ blocks defining the variables X_i, X_i^j ($j \in [\![1\,;\,q_i]\!]$), and $\bar{X}_i$, $\bar{X}_i^j$ ($j \in [\![1\,;\,\bar{q}_i]\!]$):

$$
\begin{array}{rl}
[X_i\,;\bar{X}_i] = \mathsf{var}(A_\phi^i) & \\
[X_i^1\,;U_i^2] = \mathsf{copy}(X_i) & [\bar{X}_i^1\,;\bar{U}_i^2] = \mathsf{copy}(\bar{X}_i) \\
[X_i^2\,;U_i^3] = \mathsf{copy}(U_i^2) & [\bar{X}_i^2\,;\bar{U}_i^3] = \mathsf{copy}(\bar{U}_i^2) \\
\vdots & \vdots \\
[X_i^{q_i-2}\,;U_i^{q_i-1}] = \mathsf{copy}(U_i^{q_i-2}) & \vdots \\
[X_i^{q_i-1}\,;X_i^{q_i}] = \mathsf{copy}(U_i^{q_i-1}) & [\bar{X}_i^{\bar{q}_i-2}\,;\bar{U}_i^{\bar{q}_i-1}] = \mathsf{copy}(\bar{U}_i^{\bar{q}_i-2}) \\
 & [\bar{X}_i^{\bar{q}_i-1}\,;\bar{X}_i^{\bar{q}_i}] = \mathsf{copy}(\bar{U}_i^{\bar{q}_i-1})
\end{array}
\qquad (*)
$$

  - For each $c \in [\![1\,;\quad]\!]$, let $C_c = \quad_1 \vee \quad_2 \vee \ldots \vee \quad_k$, with $k = k(C_c)$. Consider $p \in [\![1\,;\,k]\!]$. There exist integers i, j such that $\quad_p$ is the j-th occurrence of a literal x_i or $\neg x_i$, we respectively write $L_p = X_i^j$ or $L_p = \bar{X}_i^j$. We add the following $k-1$ blocks defining Γ_c:

$$
\begin{array}{c}
V_c^2 = \mathsf{or}(L_1\,;L_2) \\
V_c^3 = \mathsf{or}(V_c^2\,;L_3) \\
\vdots \\
V_c^{k-1} = \mathsf{or}(V_c^{k-2}\,;L_{k-1}) \\
\Gamma_c = \mathsf{or}(V_c^{k-1}\,;L_k)
\end{array}
\qquad (**)
$$

  - Since $= C_1 \wedge C_2 \wedge \ldots \wedge C_\gamma$, the formula variable A_ϕ is defined by the

following $\quad -1$ blocks:

$$\begin{aligned} W_2 &= \mathsf{and}(\Gamma_1;\Gamma_2) \\ W_3 &= \mathsf{and}(W_2;\Gamma_3) \\ &\vdots \\ W_{\gamma-1} &= \mathsf{and}(W_{\gamma-2};\Gamma_{\gamma-1}) \\ A_\phi &= \mathsf{and}(W_{\gamma-1};\Gamma_\gamma) \end{aligned} \qquad (***)$$

– The m copies $A_\phi^1;:::;A_\phi^m$ of A_ϕ are defined with the following $m-1$ blocks:

$$\begin{aligned} [A_\phi^1;Y_2] &= \mathsf{copy}(A_\phi) \\ [A_\phi^2;Y_3] &= \mathsf{copy}(Y_2) \\ &\vdots \\ [A_\phi^{m-2};Y_{m-1}] &= \mathsf{copy}(Y_{m-2}) \\ [A_\phi^{m-1};A_\phi^m] &= \mathsf{copy}(Y_{m-1}) \end{aligned} \qquad (****)$$

3.4. The Main Result. THEOREM 3.14. *Let be a boolean formula, and I_ϕ the 3DT-instance defined in Section 3.3. The construction of I_ϕ is polynomial in the size of , and is satisfiable iff I_ϕ is 3DT-collapsible.*

Proof. The polynomial time complexity of the construction of I_ϕ is trivial. We use the same notations as in the construction, with $\mathcal{B}$ the block decomposition of I_ϕ. One can easily check that each variable in $(*)$, $(**)$, $(***)$, and $(****)$ has exactly one source block and one target block. Then, by Lemma 3.13, we know that $(I_\phi;\mathcal{B})$ is a valid context, and remains so after any number of 3DT-steps, hence Properties 3.8, 3.9, 3.10, and 3.11 are satisfied by respectively each block copy, and, or and var, of I_ϕ.

$\boxed{\Rightarrow}$ Assume first that is satisfiable. Consider a truth assignment satisfying : let P be the set of indices $i \in [\![1\,;m]\!]$ such that x_i is assigned to true. Starting from I_ϕ, we can follow a path of 3DT-steps that activates all the variables of I_ϕ in the following order:

- For $i \in [\![1\,;m]\!]$, if $i \in P$, activate X_i in the corresponding block var in $(*)$. Then, with the blocks copy, activate successively all intermediate variables U_i^j for $j = 2$ to $q_i - 1$, and variables X_i^j for $j \in [\![1\,;q_i]\!]$.
  Otherwise, if $i \notin P$, activate $\bar{X}_i$, all intermediate variables $\bar{U}_i^j$ for $j = 2$ to $\bar{q}_i - 1$, and the variables $\bar{X}_i^j$ for $j \in [\![1\,;\bar{q}_i]\!]$
- For each $c \in [\![1\,; \quad]\!]$, let $C_c = \ _1 \vee \ _2 \vee ::: \vee \ _k$, with $k = k(C_c)$. Since C_c is true with the selected truth assignment, at least one literal $\ _{p_0}$, $p_0 \in [\![1\,;k]\!]$, is true. If $\ _{p_0}$ is the j-th occurrence of a literal x_i or $\neg x_i$, then the corresponding variable L_{p_0} ($L_{p_0} = X_i^j$ or $L_{p_0} = \bar{X}_i^j$) has been activated previously. Using the blocks or in $(**)$, we activate successively each intermediate variable V_c^p for $p = p_0;:::;k-1$, and finally we activate the variable Γ_c.
- Since all variables Γ_c, $c \in [\![1\,; \quad]\!]$, have been activated, using the blocks and in $(***)$, we activate each intermediate variable W_c for $c = 2$ to $c = \quad -1$, and the formula variable A_ϕ.

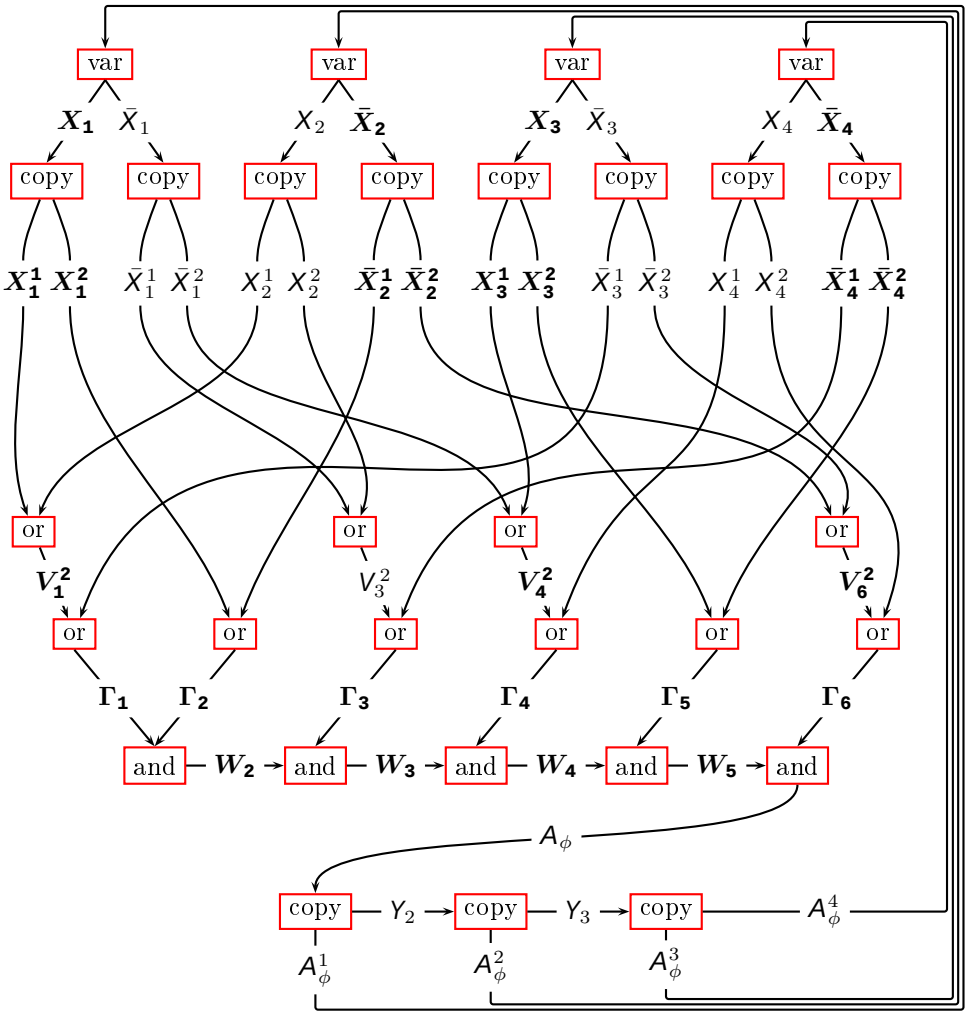


Fig. 3.9: Schematic diagram of the blocks defining I_ϕ for $\quad = (x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee x_4) \wedge (x_3 \vee \neg x_4) \wedge (\neg x_2 \vee \neg x_3 \vee x_4)$. For each variable, we draw an arc between its source and target block. Note that $\quad$ is satisfiable (e.g., with the assignment $x_1 = x_3 = \mathit{true}$ and $x_2 = x_4 = \mathit{false}$). A set of variables that can be activated before A_ϕ is in bold, they correspond to the terms being true in $\quad$ for the assignment $x_1 = x_3 = \mathit{true}$ and $x_2 = x_4 = \mathit{false}$.

- With the blocks copy in $(****)$, we activate successively all the intermediate variables Y_i, $i \in [\![2\,;\,m-1]\!]$ and the m copies $A_\phi^1;\cdots;A_\phi^m$ of A_ϕ.
- For $i \in [\![1\,;\,m]\!]$, since the variable A_ϕ^i has been activated, we activate in the block var of $(*)$ the remaining variable X_i or $\bar{X}_i$. We also activate all its copies and corresponding intermediate variables U_i^j or $\bar{U}_i^j$.
- For $c \in [\![1\,;\quad]\!]$, in $(**)$, since all variables L_p have been activated, we activate the remaining intermediate variables V_c^p.

- At this point, every variable has been activated. Using again Lemma 3.13, we know that the resulting instance is 3DT-collapsible, and can be reduced down to the empty 3DT-instance ε.

Hence I_ϕ is 3DT-collapsible.

$\boxed{\Leftarrow}$ Assume now that I_ϕ is 3DT-collapsible: we consider a sequence of 3DT-steps reducing I_ϕ to ε. This sequence gives a total order on the set of variables: the order in which they are activated. We write Q for the set of variables activated before A_ϕ, and $P \subseteq [\![1\,;\,m]\!]$ for the set of indices i such that $X_i \in Q$ (see the variables in bold in Figure 3.9). We show that the truth assignment defined by $(x_i = \mathit{true} \Leftrightarrow i \in P)$ satisfies the formula .

- For each $i \in [\![1\,;\,m]\!]$, A^i_ϕ cannot belong to Q, using the property of the block copy in $(****)$ (each A^i_ϕ can only be activated after A_ϕ). Hence, with the block var in $(*)$, we have $X_i \in Q \vee \bar{X}_i \in Q$. Moreover, with the block copy, we have

$$\forall 1 \le j \le q_i; \quad X^j_i \in Q \Rightarrow X_i \in Q \tag{a}$$

$$\forall 1 \le j \le \bar{q}_i; \quad \bar{X}^j_i \in Q \Rightarrow \bar{X}_i \in Q \Rightarrow X_i \in Q \tag{b}$$

- Since A_ϕ is defined in a block $A_\phi = \mathsf{and}(W_{\gamma-1}; \Gamma_\gamma)$ in $(***)$, we necessarily have $W_{\gamma-1} \in Q$ and $\Gamma_\gamma \in Q$. Likewise, since $W_{\gamma-1}$ is defined by $W_{\gamma-1} = \mathsf{and}(W_{\gamma-2}; \Gamma_{\gamma-1})$, we also have $W_{\gamma-2} \in Q$ and $\Gamma_{\gamma-1} \in Q$. Applying this reasoning recursively, we have $\Gamma_c \in Q$ for each $c \in [\![1\,;\quad]\!]$.
- For each $c \in [\![1\,;\quad]\!]$, consider the clause $C_c = \quad_1 \vee \quad_2 \vee \cdots \vee \quad_k$, with $k = k(C_c)$. Using the property of the block or in $(**)$, there exists some $p_0 \in [\![1\,;k]\!]$ such that the variable L_{p_0} is activated before Γ_c: hence $L_{p_0} \in Q$. If the corresponding literal $\quad_{p_0}$ is the j-th occurrence of x_i (respectively, $\neg x_i$), then $L_{p_0} = X^j_i$ (resp., $L_{p_0} = \bar{X}^j_i$), thus by (a) (resp. (b)), $X_i \in Q$ (resp., $X_i \in Q$), and consequently $i \in P$ (resp., $i \in P$). In both cases, the literal $\quad_{p_0}$ is true in the truth assignment defined by $(x_i = \mathit{true} \Leftrightarrow i \in P)$.

If I_ϕ is 3DT-collapsible, we have found a truth assignment such that at least one literal is true in each clause of the formula , and thus is satisfiable. □

4. Sorting by Transpositions Is NP-Hard. As noted previously, there is no guarantee that any 3DT-instance I has an equivalent permutation . However, with the following theorem, we show that such a permutation can be found in the special case of assemblings of basic blocks, which is the case we are interested in, in order to complete our reduction.

THEOREM 4.1. *Let I be a 3DT-instance of span n with $\mathcal{B}$ an I-block decomposition such that $(I; \mathcal{B})$ is an assembling of basic blocks. Then there exists a permutation $\quad_I$, computable in polynomial time in n, such that $I \sim \quad_I$.*

An example of the construction of $\quad_I$ for the 3DT-instance defined in Example 3.2 is given in Figure 4.1.

The rough idea of the proof is as follows. Each block $\mathcal{B}_h$ of $\mathcal{B}$ (corresponding to positions $[\![s_h + 1\,;\,t_h]\!]$) is assigned a unique interval of integers $[\![p_h + 1\,;\,q_h]\!]$. Then we create a permutation of $[\![p_h + 1\,;\,q_h]\!]$, depending on the kind of the block, and $\quad_I$ is obtained as the concatenation of these permutations. However, external triples (in variables) need a special treatment: when a variable is activated, exactly three integers are moved from one block to another. Hence, for each variable, some integers which should appear in the target block are originally present in the source block. In other words, the part of $\quad_I$ corresponding to each block has three extra integers

for each output variable, and three missing integers for each input variable. We keep track of elements which are displaced with two functions, and : for a variable A, the three affected integers are $(A)+1$, $(A)+2$ and $(A)+1$.

Proof. Let $\mathcal{A}$ be the set of variables of the l-block decomposition $\mathcal{B}$ of $I = \langle \Sigma; T; \rangle$. Let n be the span of I, and L its domain. Note that $L = [\![1\,;\,n]\!]$. For any $h \in [\![1\,;\,l]\!]$, we write $ni(\mathcal{B}_h)$ (resp. $no(\mathcal{B}_h)$) for the number of input (resp. output) variables of $\mathcal{B}_h$. We also define two integers $p_h; q_h$ by:

$$\begin{aligned} p_1 &= 0 \\ \forall h \in [\![1\,;\,l]\!]; q_h &= p_h + t_h - s_h + 3(ni(\mathcal{B}_h) - no(\mathcal{B}_h)) \\ \forall h \in [\![2\,;\,l]\!]; p_h &= q_{h-1} \end{aligned}$$

The permutation $_I$ will be defined such that p_h and q_h have the following property for any $h \in [\![1\,;\,l]\!]$: $_I(s_h) = p_h$, and $_I(t_h) = q_h$.

We also define two applications ; over the set $\mathcal{A}$ of variables. The permutation $_I$ will be defined so that, for any variable $A = [(a; b; c); (x; y; z)]$, we have $_I((a) - 1) = (A)$ and $_I((z) - 1) = (A)$. In order to have this property, and are defined as follows.

For each $h \in [\![1\,;\,l]\!]$:

- If $\mathcal{B}_h$ is a block of the kind $[A_1; A_2] = \mathsf{copy}(A)$, define

$$(A) = p_h; \quad (A) = p_h + 4;$$

- If $\mathcal{B}_h$ is a block of the kind $A = \mathsf{and}(A_1; A_2)$, define

$$(A_1) = p_h; \quad (A_1) = p_h + 7; \quad (A_2) = p_h + 3; \quad (A_2) = p_h + 9;$$

- If $\mathcal{B}_h$ is a block of the kind $A = \mathsf{or}(A_1; A_2)$, define

$$(A_1) = p_h; \quad (A_1) = p_h + 13; \quad (A_2) = p_h + 3; \quad (A_2) = p_h + 16;$$

- If $\mathcal{B}_h$ is a block of the kind $[A_1; A_2] = \mathsf{var}(A)$, define

$$(A) = p_h + 5; \quad (A) = p_h + 9;$$

Note that for every $A \in \mathcal{A}$, (A) and (A) are defined once and only once, depending on the kind of the block $\mathcal{B}_{target(A)}$. As already noted, the permutation $_I$ is designed in such a way that the image by $_I$ of an interval $[\![s_h + 1\,;\,t_h]\!]$ is essentially the interval $[\![p_h + 1\,;\,q_h]\!]$. However, there are exceptions: namely, for each variable A, the integers $(A)+1$; $(A)+2$; $(A)+1$, which are included in $[\![p_{target(A)} + 1\,;\,q_{target(A)}]\!]$, are in the image of $[\![s_{source(A)} + 1\,;\,t_{source(A)}]\!]$. This is formally described as follows. For each $h \in [\![1\,;\,k]\!]$ we define a set P_h by:

$$\begin{aligned} P_h = [\![p_h + 1\,;\,q_h]\!] \cup & \bigcup_{A \text{ output of } \mathcal{B}_h} \{ (A)+1; \ (A)+2; \ (A)+1\} \\ & - \bigcup_{A \text{ input of } \mathcal{B}_h} \{ (A)+1; \ (A)+2; \ (A)+1\} \end{aligned}$$

We note that the sets $\{ (A)+1; \ (A)+2; \ (A)+1\}$ are disjoint for different variables A, and are each included in their respective interval $[\![p_{target(A)} + 1\,;\,q_{target(A)}]\!]$. Hence for any $h \in [\![1\,;\,l]\!]$, we have $|P_h| = q_h - p_h + 3no(\mathcal{B}_h) - 3ni(\mathcal{B}_h) = t_h - s_h$. Moreover, the sets P_h, $h \in [\![1\,;\,l]\!]$, form a partition of the set $[\![1\,;\,n]\!]$.

We can now create the permutation σ_I. The image of 0 is 0, and for each h_0 from 1 to l, we define the restriction of σ_I over $[\![s_{h_0}+1\,;\,t_{h_0}]\!]$ as a permutation of P_{h_0}, with the constraint that $\sigma_I(t_{h_0}) = q_{h_0}$. Note that, if this condition is fulfilled, then we can assume $\sigma_I(s_{h_0}) = p_{h_0}$, since, if $h_0 = 1$, $\sigma_I(s_1) = \sigma_I(0) = 0 = p_1$, and if $h_0 > 1$, $\sigma_I(s_{h_0}) = \sigma_I(t_{h_0-1}) = q_{h_0-1} = p_{h_0}$.

The definition of σ_I over each kind of block is given in Table 4.1. This table is obtained by applying the following rules, until $\sigma_I(u)$ is defined for all $u \in [\![s_{h_0}+1\,;\,t_{h_0}]\!]$.

For all input variable of $\mathcal{B}_{h_0}$ $A = [(a;b;c);(x;y;z)]$;

$$\sigma_I(\ (z)) = \ (A) + 3 \qquad (R_1)$$

$$\sigma_I(\ (c)) = \ (A) + 2 \qquad (R_2)$$

For all output variable of $\mathcal{B}_{h_0}$ $A = [(a;b;c);(x;y;z)]$;

$$\sigma_I(\ (x)) = \ (A) + 1 \qquad (R_3)$$

$$\sigma_I(\ (b)) = \ (A) + 1 \qquad (R_4)$$

$\forall u \in [\![s_{h_0}+1\,;\,t_{h_0}]\!]$ such that $succ_{I_\phi}^{-1}(u) \in [\![s_{h_0}+1\,;\,t_{h_0}]\!]$

$$\sigma_I(u) = \sigma_I(succ_I^{-1}(u) - 1) + 1 \qquad (R_5)$$

We can see in Table 4.1 that rules (R_1) and (R_2) indeed apply to every input variable, and rules (R_3) and (R_4) apply to every output variable. Moreover:

$$\begin{array}{c}\text{Rule } (R_5) \text{ applies to every } u \in [\![s_{h_0+1}\,;\,t_{h_0}]\!] \text{ such that} \\ u \in \{\ (b);\ (c);\ (x);\ (z) \mid A = [(a;b;c);(x;y;z)] \text{ is input/output of } \mathcal{B}_{h_0}\}:\end{array} \qquad (P_1)$$

A simple case by case analysis shows that the following properties are also satisfied:

$$\sigma_I \text{ defines a bijection from } [\![s_{h_0}+1\,;\,t_{h_0}]\!] \text{ to } P_{h_0} \text{ such that } \sigma_I(t_{h_0}) = q_{h_0} \qquad (P_2)$$

For all input variable of $\mathcal{B}_{h_0}$ $A = [(a;b;c);(x;y;z)]$;

$$\sigma_I(\ (a) - 1) = \ (A) \qquad (P_3)$$

$$\sigma_I(\ (z) - 1) = \ (A) \qquad (P_4)$$

For all output variable of $\mathcal{B}_{h_0}$ $A = [(a;b;c);(x;y;z)]$;

$$\sigma_I(\ (y) - 1) = \ (A) + 2 \qquad (P_5)$$

$$\sigma_I(\ (b) - 1) = \ (A) + 1 \qquad (P_6)$$

Now that we have defined the permutation σ_I, we need to show that σ_I is equivalent to I. Following Definition 2.8, we have $\sigma_I(0) = 0$. Then, $L = [\![1\,;\,n]\!]$, so let us fix any $u \in [\![1\,;\,n]\!]$, and verify that $\sigma_I(u) = \sigma_I(succ_I^{-1}(u) - 1) + 1$. Let h be the integer such that $u \in [\![s_h+1\,;\,t_h]\!]$.

First, consider the most general case, and assume that there is no variable $A = [(a;b;c);(x;y;z)]$ such that $u \in \{\ (b);\ (c);\ (x);\ (z)\}$. Note that this case includes $u = \ (d)$, where d is part of some internal triple. Then, by Property (P_1), we know that Rule (R_5) applies to u, hence we directly have $\sigma_I(u) = \sigma_I(succ_I^{-1}(u) - 1) + 1$.

Suppose now that there exists some variable $A = [(a;b;c);(x;y;z)]$ such that $u \in \{\ (b);\ (c);\ (x);\ (z)\}$. Then Rules (R_1) and (R_2), and Properties (P_3) and (P_4) apply in the target block of A. Also, Rules (R_3) and (R_4), and Properties (P_5)

Table 4.1: Definition of $_I$ over an interval $[\![\mathsf{s}_{h_0}+1\,;\,\mathsf{t}_{h_0}]\!]$, where $\mathcal{B}_{h_0}$ is one of the blocks copy, and, or, var. We write $\mathsf{s}=\mathsf{s}_{h_0}$ and $\mathsf{p}=\mathsf{p}_{h_0}$. We give the line $^{-1}(u)$ as a reminder of the definition of each block. We also add a column for $u=\mathsf{s}$ as a reminder of the fact that $_I(\mathsf{s})=\mathsf{p}$.

- If $\mathcal{B}_{h_0}$ is a block of the kind $[A_1;A_2]=\mathsf{copy}(A)$, we write $_1;\ _1;\ _2;\ _2$ for the respective values of $(A_1);\ (A_1);\ (A_2);\ (A_2)$.

u =	s	$\mathsf{s}+1$	$\mathsf{s}+2$	$\mathsf{s}+3$	$\mathsf{s}+4$	$\mathsf{s}+5$	$\mathsf{s}+6$	$\mathsf{s}+7$	$\mathsf{s}+8$	$\mathsf{s}+9$
$_I(u)=$	p	$_1+2$	$\mathsf{p}+8$	$\mathsf{p}+4$	$\mathsf{p}+3$	$_2+2$	$\mathsf{p}+7$	$_1+1$	$_1+1$	$\mathsf{p}+6$
$^{-1}(u)=$		a	y_1	e	z	d	y_2	x_1	b_1	c

u =	$\mathsf{s}+10$	$\mathsf{s}+11$	$\mathsf{s}+12$
$_I(u)=$	$_2+1$	$_2+1$	$\mathsf{p}+9$
$^{-1}(u)=$	x_2	b_2	f

- If $\mathcal{B}_{h_0}$ is a block of the kind $A=\mathsf{and}(A_1;A_2)$, we write $\ ;\ $ for the respective values of $(A);\ (A)$.

u =	s	$\mathsf{s}+1$	$\mathsf{s}+2$	$\mathsf{s}+3$	$\mathsf{s}+4$	$\mathsf{s}+5$	$\mathsf{s}+6$	$\mathsf{s}+7$	$\mathsf{s}+8$	$\mathsf{s}+9$
$_I(u)=$	p	$\mathsf{p}+14$	$\mathsf{p}+7$	$\mathsf{p}+3$	$\mathsf{p}+13$	$\mathsf{p}+9$	$\mathsf{p}+6$	$+2$	$\mathsf{p}+12$	$\mathsf{p}+11$
$^{-1}(u)=$		a_1	e	z_1	a_2	c_1	z_2	d	y	c_2

u =	$\mathsf{s}+10$	$\mathsf{s}+11$	$\mathsf{s}+12$
$_I(u)=$	$+1$	$+1$	$\mathsf{p}+15$
$^{-1}(u)=$	x	b	f

- If $\mathcal{B}_{h_0}$ is a block of the kind $A=\mathsf{or}(A_1;A_2)$, we write $\ ;\ $ for the respective values of $(A);\ (A)$.

u =	s	$\mathsf{s}+1$	$\mathsf{s}+2$	$\mathsf{s}+3$	$\mathsf{s}+4$	$\mathsf{s}+5$	$\mathsf{s}+6$	$\mathsf{s}+7$	$\mathsf{s}+8$	$\mathsf{s}+9$
$_I(u)=$	p	$\mathsf{p}+7$	$\mathsf{p}+13$	$\mathsf{p}+3$	$\mathsf{p}+9$	$+2$	$\mathsf{p}+12$	$\mathsf{p}+11$	$+1$	$+1$
$^{-1}(u)=$		a_1	b'	z_1	a_2	d	y	a'	x	b

u =	$\mathsf{s}+10$	$\mathsf{s}+11$	$\mathsf{s}+12$	$\mathsf{s}+13$	$\mathsf{s}+14$	$\mathsf{s}+15$
$_I(u)=$	$\mathsf{p}+16$	$\mathsf{p}+6$	$\mathsf{p}+15$	$\mathsf{p}+10$	$\mathsf{p}+8$	$\mathsf{p}+18$
$^{-1}(u)=$	f	z_2	c_1	e	c'	c_2

- If $\mathcal{B}_{h_0}$ is a block of the kind $[A_1;A_2]=\mathsf{var}(A)$, we write $_1;\ _1;\ _2;\ _2$ for the respective values of $(A_1);\ (A_1);\ (A_2);\ (A_2)$.

u =	s	$\mathsf{s}+1$	$\mathsf{s}+2$	$\mathsf{s}+3$	$\mathsf{s}+4$	$\mathsf{s}+5$	$\mathsf{s}+6$	$\mathsf{s}+7$	$\mathsf{s}+8$	$\mathsf{s}+9$
$_I(u)=$	p	$_1+2$	$\mathsf{p}+5$	$\mathsf{p}+3$	$_2+2$	$\mathsf{p}+12$	$\mathsf{p}+1$	$\mathsf{p}+14$	$\mathsf{p}+4$	$_1+1$
$^{-1}(u)=$		d_1	y_1	a	d_2	y_2	e_1	a'	e_2	x_1

u =	$\mathsf{s}+10$	$\mathsf{s}+11$	$\mathsf{s}+12$	$\mathsf{s}+13$	$\mathsf{s}+14$	$\mathsf{s}+15$	$\mathsf{s}+16$	$\mathsf{s}+17$	$\mathsf{s}+18$
$_I(u)=$	$_1+1$	$\mathsf{p}+13$	$\mathsf{p}+9$	$\mathsf{p}+8$	$\mathsf{p}+2$	$\mathsf{p}+11$	$_2+1$	$_2+1$	$\mathsf{p}+15$
$^{-1}(u)=$	b_1	f_1	c'	z	b'	c	x_2	b_2	f_2

$[X_1, X_2] = \mathsf{var}(Y)$ $\qquad$ $Y = \mathsf{or}(X_1, X_2)$

Input variable (target of): Y $\qquad$ Input variables (target of): X_1, X_2
Output variables (source of): X_1, X_2 $\qquad$ Output variable (source of): Y

$s_1 = 0 \qquad t_1 = 18$ $\qquad$ $s_2 = 18 \qquad t_2 = 33$
$p_1 = 0 \qquad q_1 = p_1 + 18 - 3 = 15$ $\qquad$ $p_2 = 15 \qquad q_2 = p_2 + 15 + 3 = 33$

$\alpha(Y) = p_1 + 5 = 5 \qquad \alpha(X_1) = p_2 = 15 \qquad \alpha(X_2) = p_2 + 3 = 18$
$\beta(Y) = p_1 + 9 = 9 \qquad \beta(X_1) = p_2 + 13 = 28 \qquad \beta(X_2) = p_2 + 16 = 31$

Definition of π_I over $[\![s_1 + 1\,;\, t_1]\!]$:

u	$=$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	…
$\pi_I(u)$	$=$	0	17	5	3	20	12	1	14	4	29	16	13	9	8	2	11	32	19	15	…
$\psi^{-1}(u)$	$=$	d_1	y_1	a	d_2	y_2	e_1	a'	e_2	x_1	b_1	f_1	c'	z	b'	c	x_2	b_2	f_2	…	

over $[\![s_2 + 1\,;\, t_2]\!]$:

u	$=$	…	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
$\pi_I(u)$	$=$	…	22	28	18	24	7	27	26	10	6	31	21	30	25	23	33
$\psi^{-1}(u)$	$=$	…	a_1	b''	z_1	a_2	d	y	a''	x	b	f	z_2	c_1	e	c''	c_2

Fig. 4.1: Creation of a permutation σ_I equivalent to the assembling of basic blocks $I = \langle \Sigma; T; \ \rangle$ of span 33 defined in Example 3.2, using the proof of Theorem 4.1.

and (P_6) apply in the source block of A. Combining all these equations together, we have:

$$\begin{aligned}
\sigma_I(\ \ (b)) &= \ \ (A) + 1 = \ \ \sigma_I(\ \ (a) - 1) + 1 && \text{by } (R_3) \text{ and } (P_3)\\
\sigma_I(\ \ (c)) &= \ \ (A) + 2 = \ \ \sigma_I(\ \ (b) - 1) + 1 && \text{by } (R_2) \text{ and } (P_5)\\
\sigma_I(\ \ (x)) &= \ \ (A) + 1 = \ \ \sigma_I(\ \ (z) - 1) + 1 && \text{by } (R_4) \text{ and } (P_4)\\
\sigma_I(\ \ (z)) &= \ \ (A) + 3 = \ \ \sigma_I(\ \ (y) - 1) + 1 && \text{by } (R_1) \text{ and } (P_6)
\end{aligned}$$

For $u = \ \ (b)$ (resp. $\ \ (c)$; $\ \ (x)$; $\ \ (z)$), we have $succ_I^{-1}(u) = \ \ (a)$ (resp. $\ \ (b)$; $\ \ (z)$; $\ \ (y)$). Hence, in all four cases, we have $\sigma_I(u) = \ \ \sigma_I(succ_I^{-1}(u) - 1) + 1$, which completes the proof that σ_I is equivalent to I.
□

With the previous theorem, we now have all the necessary ingredients to prove the main result of this paper.

THEOREM 4.2. *The* SORTING BY TRANSPOSITIONS *problem is NP-hard.*

Proof. The reduction from SAT is as follows: given any instance $\ \ $ of SAT, create a 3DT-instance I_ϕ, being an assembling of basic blocks, which is 3DT-collapsible iff $\ \ $ is satisfiable (Theorem 3.14). Then create a 3-permutation σ_{I_ϕ} equivalent to I_ϕ (Theorem 4.1). The above two steps can be done in polynomial time. Finally, set $k = d_b(\ \ \sigma_{I_\phi})$=3 $= n$=3. We then have:

$$\begin{aligned}
&\ \ \text{is satisfiable} \Leftrightarrow I_\phi \text{ is 3DT-collapsible}\\
&\qquad \Leftrightarrow d_t(\ \ \sigma_{I_\phi}) = k \text{ (by Theorem 2.12, since } \sigma_{I_\phi} \sim I_\phi)\\
&\qquad \Leftrightarrow d_t(\ \ \sigma_{I_\phi}) \leq k \text{ (by Property 1.7):}
\end{aligned}$$

□

COROLLARY 4.3. *The following decision problem from [10] is NP-hard: given a permutation of $[\![0\,;n]\!]$, is the equality $d_t(\ ) = d_b(\ )$=3 satisfied?*

Conclusion. In this paper, we have proved that the SORTING BY TRANSPOSITIONS problem is NP-hard, thus answering a long-standing open question. However, a number of questions remain open. For instance, does this problem admit a polynomial-time approximation scheme? We note that the reduction we have provided does not answer this question, since it is not a gap-preserving reduction. Indeed, in our reduction, if a formula is not satisfiable, it can be seen that we have $d_t(\ _{I_\phi}) = d_b(\ _{I_\phi})$=3 + 1.

Also, do there exist some relevant parameters for which the problem is fixed parameter tractable? A parameter that comes to mind when dealing with the transposition distance is the length of the exchanged factors (i.e., the value $\max\{j-i;k-j\}$ for a transposition $_{i,j,k}$). Does the problem become tractable if this parameter is bounded? In fact, the answer to this question is negative if only the length of the smallest factor, $\min\{j-i;k-j\}$, is bounded: in our reduction, this parameter is upper bounded by 6 for every transposition needed to sort $_{I_\phi}$, independently of the formula .

REFERENCES

[1] A. Amir, Y. Aumann, G. Benson, A. Levy, O. Lipsky, E. Porat, S. Skiena, and U. Vishne. Pattern matching with address errors: Rearrangement distances. *J. Comput. Syst. Sci.*, 75(6):359–370, 2009.

[2] A. Amir, Y. Aumann, P. Indyk, A. Levy, and E. Porat. Efficient computations of ℓ_1 and ℓ_∞ rearrangement distances. In Nivio Ziviani and Ricardo A. Baeza-Yates, editors, *SPIRE*, volume 4726 of *Lecture Notes in Computer Science*, pages 39–49. Springer, 2007.

[3] V. Bafna and P. A. Pevzner. Sorting permutations by transpositions. In *SODA*, pages 614–623, 1995.

[4] V. Bafna and P. A. Pevzner. Sorting by transpositions. *SIAM J. Discrete Math.*, 11(2):224–240, 1998.

[5] M. Benoît-Gagné and S. Hamel. A new and faster method of sorting by transpositions. In B. Ma and K. Zhang, editors, *CPM*, volume 4580 of *Lecture Notes in Computer Science*, pages 131–141. Springer, 2007.

[6] D. Bongartz. *Algorithmic Aspects of Some Combinatorial Problems in Bioinformatics*. PhD thesis, RWTH Aachen University, Germany, 2006.

[7] L. Bulteau, G. Fertin, and I. Rusu. Sorting by transpositions is difficult. In L. Aceto, M. Henzinger, and J. Sgall, editors, *ICALP (1)*, volume 6755 of *Lecture Notes in Computer Science*, pages 654–665. Springer, 2011.

[8] B. Chitturi and I. H. Sudborough. Bounding prefix transposition distance for strings and permutations. In *HICSS*, page 468. IEEE Computer Society, 2008.

[9] D. A. Christie. Sorting permutations by block-interchanges. *Inf. Process. Lett.*, 60(4):165–169, 1996.

[10] D. A. Christie. *Genome Rearrangement Problems*. PhD thesis, University of Glasgow, Scotland, 1998.

[11] D. A. Christie and R. W. Irving. Sorting strings by reversals and by transpositions. *SIAM J. Discrete Math.*, 14(2):193–206, 2001.

[12] G. Cormode and S. Muthukrishnan. The string edit distance matching problem with moves. In *SODA*, pages 667–676, 2002.

[13] Z. Dias and J. Meidanis. Sorting by prefix transpositions. In A. H. F. Laender and A. L. Oliveira, editors, *SPIRE*, volume 2476 of *Lecture Notes in Computer Science*, pages 65–76. Springer, 2002.

[14] I. Elias and T. Hartman. A 1.375-approximation algorithm for sorting by transpositions. *IEEE/ACM Trans. Comput. Biology Bioinform.*, 3(4):369–379, 2006.

[15] H. Eriksson, K. Eriksson, J. Karlander, L. J. Svensson, and J. Wästlund. Sorting a bridge hand. *Discrete Mathematics*, 241(1-3):289–300, 2001.

[16] J. Feng and D. Zhu. Faster algorithms for sorting by transpositions and sorting by block interchanges. *ACM Transactions on Algorithms*, 3(3), 2007.

[17] G. Fertin, A. Labarre, I. Rusu, É. Tannier, and S. Vialette. *Combinatorics of genome rearrangements*. The MIT Press, 2009.

[18] Q.-P. Gu, S. Peng, and Q. M. Chen. Sorting permutations and its applications in genome analysis. *Lectures on Mathematics in the Life Science*, 26:191–201, 1999.

[19] S. A. Guyer, L. S. Heath, and J. P. Vergara. Subsequence and run heuristics for sorting by transpositions. Technical report, Virginia State University, 1997.

[20]