The Complexity of the Evaluation of Complex Algebra Expressions

Dan Suciu*
University of Pennsylvania, USA
suciu@saul.cis.upenn.edu

Jan Paredaens
University of Antwerp, Belgium
pareda@uia.ua.ac.be

Abstract

The Abiteboul and Beeri algebra for complex objects can express a query whose meaning is transitive closure, but the algorithm naturally associated to this query needs exponential space. We show that any other query in the algebra which expresses transitive closure needs exponential space, under a "call by value" evaluation strategy. This proves that in general the powerset is an intractable operator for implementing fixpoint queries.

1 Introduction

Abiteboul and Beeri in [AB88] have shown that powerset can express transitive closure (tc), in a language for complex objects without fixpoints or any other form of iterations. But the obvious way of doing that is by a query whose naturally associated algorithm requires exponential space (and time). We prove here that in order to express tc with powerset, exponential space (and time) is indeed needed, under a "call by value" evaluation strategy.

This result is of a different nature than classical inexpressibility results (like transitive closure is not expressible in FO [AU79] or *even* is not expressible in FO+LFP), because it says that transitive closure, although expressible in a particular language, is not expressible *efficiently* in that language. This denotes a mismatch between the complexity of the natural way of computing queries in that language, and the complexity of the best Turing Machine for computing those queries. It is in the same spirit as Abiteboul and Vianu's result that *even* cannot be computed in polynomial space on a Generic Machine [AV91].

Our result depends crucially on the evaluation strategy considered, namely the *eager* (or *call-by-value*) evaluation strategy. While this strategy seems to be naturally associated with a language having powerset as a primitive, it does not exclude other strategies, to which our result no longer applies. E.g. FO+LFP can be expressed in Universal Second-Order Logic (USO), which, in turn can be embedded into the algebra with powerset. USO has the expressive power of co-NP, and its natural associated evaluation strategy has a complexity which is no more than co-NP, hence in PSPACE. This gives an example of a reasonable fragment of the algebra with powerset (namely the image of USO under the embedding) with a PSPACE evaluation strategy, which can express transitive closure.

*The author was partially supported by grant NSF CCR-90-57570

Hull and Su in [HS91] and Kuper and Vardi in [KV93] study the complexity of logics for complex objects. Carried over from logic to the algebra with powerset, their results establish a strict hierarchy of languages corresponding to the depth of nesting of the powerset [HS91], and establish a strong relationship between the expressive power of these languages and certain complexity classes [KV93]. Both results are orthogonal to ours: they concern about *what* powerset can express, while our result concerns about *how efficiently* powerset can express.

Technically, our result is slightly stronger, in that it proves that powerset cannot express efficiently *deterministic* transitive closure (see [Imm87]), i.e. transitive closure of a graph whose nodes have outdegree ≤ 1. More precisely, we prove that in order to compute the transitive closure of the relation $r_n = \{(0,1),(1,2),\ldots,(n-1,n)\}$ in a certain complex object algebra with powerset, exponential space is needed. In addition, in any query over r_n which does not require exponential space, we can replace all occurrences of powerset with some approximation expressible in that algebra without powerset.

The proof technique we use here is related to that used by Aho and Ullman [AU79] to prove that transitive closure is not expressible in the relational algebra. They identify all possible results which one can get by applying a relational algebra expression to r_n, and show that none of them is the transitive closure. Here, we identify a set of possible complex object results which one can get by applying a complex object algebra expression of polynomial complexity to r_n, and show that the transitive closure of r_n is not among them. Obviously expressions involving powerset may take us out of this set, and the hard part of the proof consists in showing that, whenever this happens, the complexity under the call-by-value evaluation strategy is exponential.

A consequence of our results is that powerset is not an efficient operator for the implementation of fixpoint queries in general. Clearly, adding *while* to the algebra, instead of powerset, gives us the same computational power but it evidently only uses polynomial time (and space) for computing transitive closure.

We conjecture that *any* query expressible efficiently with powerset is expressible also without powerset. However, this problem remains open.

In Section 2 we define the *nested relational algebra with powerset* (which has the same expressive power as Abiteboul and Beeri's *algebra*) and the complexity of its evaluation. We state our main results in Section 3. The proof follows from three facts: (1) we prove that the *abstract expressions*, defined in Section 4, are closed under application of functions in the nested relational algebra (this is shown in Section 5), (2) the relation r_n can be expressed as an abstract expression (Fact 4.9), but its transitive closure cannot (this is shown in Section 6), and finally (3) any function in the nested relational algebra with powerset applied to some abstract expression either yields another abstract expression, or has exponential complexity (this is shown in Section 7).

2 A Query Language for Complex Objects

Abiteboul and Beeri, in [AB88], define the *complex values algebra*, as a functional language for complex objects, and show that it has the same expressive power as the *domain independent calculus*, an extension of first order logic to complex objects. Powerset is explicitly included in the algebra, and the authors show how powerset can be used to compute transitive closure.

Here we define a database query language for complex objects which we call $\mathcal{NRA}$, for *nested relational algebra*, which has the same expressive power as the *algebra without powerset* in [AB88]. Similarly, $\mathcal{NRA}(\mathsf{powerset})$, the extension of $\mathcal{NRA}$ with powerset, has the same expressive power as the *complex values algebra*.

2.1 Complex Objects

The values on which $\mathcal{NRA}$ operates are the complex objects. These essentially consist of tuples or finite sets of simpler complex objects, starting from some atomic values. We will only consider **typed complex objects** in this paper. Namely we define a **type** by the grammar:

$$t ::= unit \mid \mathbb{B} \mid \mathbb{Z} \mid t \times t \mid \{t\}$$

Now we can formally define complex objects.

Definition 2.1 *A* **complex object** *is defined by the following:*

- *The empty tuple is a complex object. It is denoted by $\langle\rangle$, and its type is unit.*
- T *and* F *denote the only two complex objects of type $\mathbb{B}$.*
- *Any integer x is a complex object of type $\mathbb{Z}$. We will use decimal notation for integers.*
- *If Γ_1 and Γ_2 denote complex objects of types t_1 and t_2 respectively, then $\langle\Gamma_1, \Gamma_2\rangle$ denotes another complex object, called a* tuple, *of type $t_1 \times t_2$.*
- *If $\Gamma_1, \ldots, \Gamma_n$, $n \geq 0$, are all different and denote complex objects of type t, then $\{\Gamma_1, \ldots, \Gamma_n\}$ denotes another complex object having type $\{t\}$. Particularly, for $n = 0$ we get that the empty set $\{\}$ is a complex object of any type $\{t\}$.*

We write $\Gamma : t$ to emphasize that Γ is a complex object of type t. E.g. $\{\langle 3, 9\rangle, \langle 7, 3\rangle\} : \{\mathbb{Z} \times \mathbb{Z}\}$, $\{4, 2\} : \{\mathbb{Z}\}$, and $\{\langle 3, \{4, 2\}\rangle, \langle 9, \{\}\rangle\} : \{\mathbb{Z} \times \{\mathbb{Z}\}\}$. By contrast, $\{\langle 4, 6\rangle, 9\}$ is *not* a complex object, because it is not properly typed.

To every type t we associate a set of complex objects, namely the set of complex objects of type t. We will freely use the same notation t both for the type and for the set of complex objects of that type. Due to the fact that the empty set $\{\}$ belongs to any type $\{t\}$, the types are not disjoint. E.g. the complex object $\{\{\}, \{\{\}\}\}$ belongs to all types of the form $\{\{\{t\}\}\}$, like $\{\{\{\mathbb{Z}\}\}\}$, or $\{\{\{\mathbb{B} \times \{\mathbb{Z} \times \mathbb{B}\}\}\}\}$.

Next we define the size of a notation of a complex object C, by: $\mathsf{size}(\langle\rangle) = \mathsf{size}(\mathsf{F}) = \mathsf{size}(\mathsf{T}) = \mathsf{size}(x) = 1$, $\mathsf{size}(\langle C, C'\rangle) = 1 + \mathsf{size}(C) + \mathsf{size}(C')$ and $\mathsf{size}(\{C_1, \ldots, C_n\}) = 1 + \mathsf{size}(C_1) + \ldots + \mathsf{size}(C_n)$. In particular $\mathsf{size}(\{\}) = 1$. Note that all notations for the same complex object have the same size, that is $\mathsf{size}(\{3, 7\}) = \mathsf{size}(\{7, 3\}) = 3$.

Intuitively, $\mathsf{size}(C)$ is, up to a constant factor, the number of symbols necessary to write down the complex object C, assuming we count one symbol for every integer. This assumption is not crucial for our result in Section 3, since other reasonable choices for the size function are polynomially related to this one. Indeed, suppose we define another measure size' for which $\mathsf{size}'(x) = \lceil \log(|x| + 1) \rceil$, for every integer x, i.e. we count the number of binary or decimal digits necessary to represent some integer x. Then for any complex object C there is an isomorphic image C' such that $\mathsf{size}'(C') \leq \mathsf{size}(C')^2$. Indeed, let $x_1, \ldots, x_n$ be all integers occuring in C. Substitute them with the integers $1, 2, \ldots, n$ and call C' the resulting complex object. Obviously $\mathsf{size}'(i) = \lceil \log(i + 1) \rceil \leq n$, for $i = 1, 2, \ldots, n$, and we easily deduce by induction on the structure of C', that $\mathsf{size}'(C') \leq n \cdot \mathsf{size}(C')$. Since now $n \leq \mathsf{size}(C')$, we have that $\mathsf{size}'(C') \leq \mathsf{size}(C')^2$.

$$
\frac{}{\mathsf{id}^t : t \to t} \qquad \frac{f : t_1 \to t_2 \quad g : t_2 \to t_3}{(g \circ f) : t_1 \to t_3}
$$

$$
\frac{}{!^t : t \to \mathit{unit}} \qquad \frac{f : t \to t_1 \quad g : t \to t_2}{\langle f, g \rangle : t \to t_1 \times t_2} \qquad \frac{}{\pi_1^{t_1,t_2} : t_1 \times t_2 \to t_1} \qquad \frac{}{\pi_2^{t_1,t_2} : t_1 \times t_2 \to t_2}
$$

$$
\frac{f : t_1 \to t_2}{\mathsf{map}(f) : \{t_1\} \to \{t_2\}} \qquad \frac{}{\eta^t : t \to \{t\}} \qquad \frac{}{\mu^t : \{\{t\}\} \to \{t\}}
$$

$$
\frac{}{\Delta_{\mathsf{left}}^{t_1,t_2} : t_1 \times \{t_2\} \to \{t_1 \times t_2\}}
$$

$$
\frac{}{\emptyset^t : \mathit{unit} \to \{t\}} \qquad \frac{}{\cup^t : \{t\} \times \{t\} \to \{t\}} \qquad \frac{}{=: \mathbb{Z} \times \mathbb{Z} \to \mathbb{B}} \qquad \frac{}{\mathsf{empty}^t : \{t\} \to \mathbb{B}}
$$

$$
\frac{}{\mathsf{true} : \mathit{unit} \to \mathbb{B}} \qquad \frac{}{\mathsf{false} : \mathit{unit} \to \mathbb{B}} \qquad \frac{f : t_1 \to \mathbb{B} \quad f_1 : t_1 \to t_2 \quad f_2 : t_1 \to t_2}{\mathsf{if}\ f\ \mathsf{then}\ f_1\ \mathsf{else}\ f_2 : t_1 \to t_2}
$$

Figure 1: The definition of $\mathcal{NRA}$

The notation for complex objects we consider is only one possibility. Other notations are possible, which allows "large" complex objects to have very short notations. The main result of our paper only holds for this straightforward notation we defined here, and for its corresponding size function, or a polynomially related one.

2.2 The Language $\mathcal{NRA}(\mathsf{powerset})$

2.2.1 $\mathcal{NRA}$

Semantically, a query is a function $\varphi : t_1 \to t_2$, where t_1 and t_2 are types. To qualify as a query, the function φ has to be *generic* [CH80], i.e. invariant under isomorphisms of $\mathbb{Z}$, and computable.

Next we will formally define the query language $\mathcal{NRA}$. Expressions in this language denote queries. The distinction between expressions and queries lies in the fact that several expressions may denote the same query, while other queries may not have any notation at all in this language.

$\mathcal{NRA}$ is strongly typed. The type of some expression f has the form $t_1 \to t_2$, where t_1 and t_2 are complex object types. The rules in Figure 1 give an inductive definition of the expressions in $\mathcal{NRA}$ together with their type.

Any expression f of type $t_1 \to t_2$ denotes a query of that type. We describe below the meaning of the expressions.

id^t denotes the identity function at type t. $g \circ f$ is the composition of the functions f and g. $!^t$ is the constant function such that $\forall x, !^t(x) \stackrel{\text{def}}{=} \langle\rangle$. $\langle f, g \rangle$ denotes the pair of the functions denoted by f and g, that is $\langle f, g \rangle(x) \stackrel{\text{def}}{=} \langle f(x), g(x) \rangle$. $\pi_1^{t_1,t_2}$ and $\pi_2^{t_1,t_2}$ are the first and respectively the second projection, i.e. $\pi_1^{t_1,t_2}(\langle x, y \rangle) \stackrel{\text{def}}{=} x$, $\pi_2^{t_1,t_2}(\langle x, y \rangle) \stackrel{\text{def}}{=} y$. $\mathsf{map}(f)$ is defined by $\mathsf{map}(f)(\{x_1, \ldots, x_n\}) \stackrel{\text{def}}{=} \{f(x_1), \ldots, f(x_n)\}$: it is called *replace* (written $\rho\langle f \rangle$ in [AB88]). η^t denotes the function defined by $\eta^t(x) \stackrel{\text{def}}{=} \{x\}$, that is η^t

constructs singleton sets. μ^t denotes the flattening function, that is $\mu^t\{x_1, \cdots, x_n\} \stackrel{\mathrm{def}}{=} x_1 \cup \ldots \cup x_n$ (it is called *set-collapse* in [AB88]). Further, $\Delta^{t_1,t_2}_{\mathsf{left}}(\langle x, \{y_1, \cdots, y_n\}\rangle) \stackrel{\mathrm{def}}{=} \{\langle x, y_1\rangle, \ldots, \langle x, y_n\rangle\}$, $\emptyset^t$ is the empty set, $\cup^t$ is set union, $=(\langle x, y\rangle)$ returns T iff $x = y$, $\mathsf{empty}^t(x)$ returns T iff $x = \emptyset$. Finally, $\mathsf{if}\ f\ \mathsf{then}\ f_1\ \mathsf{else}\ f_2$ is the function g such that, $\forall x$, if $f(x)$ is true then $g(x) \stackrel{\mathrm{def}}{=} f_1(x)$ and otherwise $g(x) \stackrel{\mathrm{def}}{=} f_2(x)$.

In defining $\mathcal{NRA}$ we have followed closely [BBW92], but the language has essentially the same expressive power as other formalisms for complex objects, such as Schek and Scholl's NF^2 relational algebra [SS86], as Thomas and Fischer's algebra [TF86], and as Paredaens and Van Gucht's *nested algebra* [PG88, PG92]. To substantiate this claim, we follow [BBW92] and show in the following examples how to express some of the primitives present in the latter languages.

Example 2.2 *The database projection $\Pi^{t_1,t_2}_1 : \{t_1 \times t_2\} \to \{t_1\}$ and $\Pi^{t_1,t_2}_2 : \{t_1 \times t_2\} \to \{t_2\}$ are defined by $\mathsf{map}(\pi^{t_1,t_2}_1)$ and $\mathsf{map}(\pi^{t_1,t_2}_2)$ respectively. Cartesian product $\times^{t_1,t_2} : \{t_1\} \times \{t_2\} \to \{t_1 \times t_2\}$ can be expressed as $\mu \circ \mathsf{map}(\Delta^{t_1,t_2}_{\mathsf{right}}) \circ \Delta^{\{t_1\}\times t_2}_{\mathsf{left}}$, where $\Delta^{t_1,t_2}_{\mathsf{right}} : \{t_1\} \times t_2 \to \{t_1 \times t_2\}$ is just the "symmetric" of Δ_{left}, i.e. $\Delta^{t_1,t_2}_{\mathsf{right}} \stackrel{\mathrm{def}}{=} \mathsf{map}(\langle \pi^{t_2,t_1}_2, \pi^{t_2,t_1}_1\rangle) \circ \Delta^{t_2,t_1}_{\mathsf{left}} \circ \langle \pi^{\{t_1\},t_2}_2, \pi^{\{t_1\},t_2}_1\rangle$.*

Example 2.3 $\mathsf{not} : \mathbb{B} \to \mathbb{B}$ *can be expressed by:*

$$\mathsf{if}\ \mathsf{id}^{\mathbb{B}}\ \mathsf{then}\ \mathsf{false} \circ\, !^{\mathbb{B}}\ \mathsf{else}\ \mathsf{true} \circ\, !^{\mathbb{B}}$$

Similarly we can define $\mathsf{and}, \mathsf{or} : \mathbb{B} \times \mathbb{B} \to \mathbb{B}$.

Example 2.4 *Let $p : t \to \mathbb{B}$ be some predicate in $\mathcal{NRA}$. We can define the selection $\mathsf{select}(p) : \{t\} \to \{t\}$, with the meaning $\mathsf{select}(p)(x) = \{y \mid y \in x \wedge p(y)\}$, by:*

$$\mu^t \circ \mathsf{map}(\mathsf{if}\ p\ \mathsf{then}\ \eta^t\ \mathsf{else}\ \emptyset^t \circ\, !^t)$$

Similarly, we define the predicates $\mathsf{forall}(p) : \{t\} \to \mathbb{B}$ and $\mathsf{exists}(p) : \{t\} \to \mathbb{B}$, with the meaning $\mathsf{forall}(p)(x) = \forall y \in x.p(y)$, and $\mathsf{exists}(p)(x) = \exists y \in x.p(y)$, by:

$$\begin{aligned} \mathsf{exists}(p) &\stackrel{\mathrm{def}}{=} \mathsf{not} \circ \mathsf{empty}^t \circ \mathsf{select}(p) \\ \mathsf{forall}(p) &\stackrel{\mathrm{def}}{=} \mathsf{not} \circ \mathsf{exists}(\mathsf{not} \circ p) \end{aligned}$$

Example 2.5 *Now we can define, for every type t, the equality, membership and subset predicates $=^t : t \times t \to \mathbb{B}$ and $\mathsf{member}^t : t \times \{t\} \to \mathbb{B}$ and $\subseteq^t : \{t\} \times \{t\} \to \mathbb{B}$. Suppose we have already defined $=^t$. Then we define:*

$$\mathsf{member}^t \stackrel{\mathrm{def}}{=} \mathsf{exists}(=^t) \circ \Delta^{t,t}_{\mathsf{left}}$$

and:

$$\subseteq^t \stackrel{\mathrm{def}}{=} \mathsf{forall}(\mathsf{member}^t) \circ \Delta^{t,\{t\}}_{\mathsf{right}}$$

For $t = \{t_0\}$ we define $=^t \stackrel{\mathrm{def}}{=} (\mathsf{and} \circ \langle \subseteq^{t_0}, \supseteq^{t_0}\rangle)$ (where $\supseteq^{t_0} \stackrel{\mathrm{def}}{=} \subseteq^{t_0} \circ \langle \pi^{t_0,t_0}_2, \pi^{t_0,t_0}_1\rangle$). For $t = t_1 \times t_2, t = \mathbb{Z}, t = \mathbb{B}$, and $t = unit$, the definition of $=^t$ should be clear.

The presentation of $\mathcal{NRA}$ is designed such as to make the proof of our main result easier. However the queries expressed in $\mathcal{NRA}$ are sometimes hard to read. Therefore we will relax its syntax and introduce a more friendly notation. Namely, we will:

1. Drop the type superscripts from the primitive operations, i.e. write $\cup$ instead of $\cup^t$ etc., whenever no confusion arises.
2. Allow more liberal expressions, with variables, λ-expressions, pattern-matching, and function definitions. E.g. we will define queries like $f(r) \stackrel{\text{def}}{=} \mathsf{map}(\lambda\langle\langle x,y\rangle,z\rangle.\mathsf{if}\ x = y\ \mathsf{then}\ \{z\}\ \mathsf{else}\ \{\})(r)$ instead of the official notation for f: $\mathsf{map}(\mathsf{if}\ = \circ\langle\pi_1 \circ \pi_1, \pi_2 \circ \pi_1\rangle\ \mathsf{then}\ \eta \circ \pi_2\ \mathsf{else}\ \emptyset \circ !)$.
3. Use comprehensions. E.g. we allow definitions of queries like $g(r) \stackrel{\text{def}}{=} \{\langle x,z\rangle \mid \langle\langle x,y\rangle,z\rangle \in r, x = y\}$, instead of the official $\mu \circ (\mathsf{map}(\mathsf{if}\ = \circ\langle\pi_1 \circ \pi_1, \pi_2 \circ \pi_1\rangle\ \mathsf{then}\ \eta \circ \langle\pi_1 \circ \pi_1, \pi_2\rangle\ \mathsf{else}\ \emptyset \circ !))$.

Example 2.6 *Using the abbreviations above, we can define the queries* $\mathsf{difference} : \{t\} \times \{t\} \to \{t\}$, $\mathsf{unnest} : \{t_1 \times \{t_2\}\} \to \{t_1 \times t_2\}$, $\mathsf{nest} : \{t_1 \times t_2\} \to \{t_1 \times \{t_2\}\}$, *and relation composition* $\mathsf{compose} : \{t \times t\} \times \{t \times t\} \to \{t \times t\}$ *by:*

$$\begin{aligned}
\mathsf{difference}(\langle x,y\rangle) &\stackrel{\text{def}}{=} \{z \mid z \in x; \mathsf{not}(\mathsf{member}(\langle z,y\rangle))\} \\
\mathsf{unnest}(x) &\stackrel{\text{def}}{=} \{\langle u,w\rangle \mid \langle u,v\rangle \in x; w \in v\} \\
\mathsf{nest}(x) &\stackrel{\text{def}}{=} \{\langle u, \{v \mid \langle u,v\rangle \in x\}\rangle \mid \langle u,v'\rangle \in x\} \\
\mathsf{compose}(\langle x,y\rangle) &\stackrel{\text{def}}{=} \{\langle u,w\rangle \mid \langle u,v\rangle \in x, \langle v,w\rangle \in y\}
\end{aligned}$$

We refer to [BBW92, Won93] for a detailed discussion of presentations equivalent to $\mathcal{NRA}$.

Two main results are known for $\mathcal{NRA}$:

Proposition 2.7 *[BBW92] $\mathcal{NRA}$ has the same expressive power as the algebra without powerset of [AB88].*

Proposition 2.8 *[BBW92] All queries expressible in $\mathcal{NRA}$ are in PTIME.*

So any query whose complexity is not in PTIME cannot be expressed in $\mathcal{NRA}$. But more, there are PTIME queries which *cannot* be expressed in $\mathcal{NRA}$. Namely consider the queries *transitive closure* $\mathsf{tc} : \{\mathbb{Z} \times \mathbb{Z}\} \to \{\mathbb{Z} \times \mathbb{Z}\}$, and $\mathsf{parity} : \{\mathbb{Z}\} \to \mathbb{B}$, with the meaning:

$$\begin{aligned}
\mathsf{tc}(r) &\stackrel{\text{def}}{=} \{\langle u,v\rangle \mid \exists n \geq 1, \exists u_1, \ldots, u_n.u = u_1, u_n = v, \forall i.1 \leq i < n, \langle u_i, u_{i+1}\rangle \in r\} \\
\mathsf{parity}(x) &\stackrel{\text{def}}{=} \mathsf{T} \text{ iff } \mathsf{card}(x) \text{ is even}
\end{aligned}$$

Throughout the paper, $\mathsf{card}(x)$ denotes the cardinality of the set x (i.e. the number of its elements).

Proposition 2.9 *[PG92, Won92] The queries* tc *and* parity *cannot be expressed in $\mathcal{NRA}$.*

$$\overline{\mathsf{powerset}^t : \{t\} \to \{\{t\}\}}$$

Figure 2: The definition of powerset.

2.2.2 $\mathcal{NRA}(\mathsf{powerset})$

Now we consider for every type t a new primitive operation, $\mathsf{powerset}^t$, see Figure 2. Its meaning is:

$$\mathsf{powerset}^t(x) \stackrel{\text{def}}{=} \{y \mid y \subseteq x\}$$

We denote with $\mathcal{NRA}(\mathsf{powerset})$ the language $\mathcal{NRA}$ extended with $\mathsf{powerset}^t$, for all types t. While all queries expressible in $\mathcal{NRA}$ are in PTIME, $\mathcal{NRA}(\mathsf{powerset})$ can obviously express exponential queries. More interestingly, $\mathcal{NRA}(\mathsf{powerset})$ can express PTIME queries, which are not expressible in $\mathcal{NRA}$. Following Abiteboul and Beeri [AB88], we show below how to express transitive closure in $\mathcal{NRA}(\mathsf{powerset})$.

Example 2.10 *Transitive closure* $\mathsf{tc} : \{\mathbb{Z} \times \mathbb{Z}\} \to \{\mathbb{Z} \times \mathbb{Z}\}$ *can be expressed in* $\mathcal{NRA}(\mathsf{powerset})$. *Indeed, in order to compute* $\mathsf{tc}(r)$ *for some binary relation* r*, perform the following steps:*

1. *Compute the set of values mentioned in* r, $v = \Pi_1(r) \cup \Pi_2(r)$.
2. *Compute the set of* all *binary relations on* v, $A = \mathsf{powerset}(v \times v)$.
3. *Select those which are transitively closed and contain* r, *i.e.* $B = \mathsf{select}(p)(A)$. *Here* p *is the predicate* $p(x) = (r \subseteq x) \wedge (\mathsf{compose}(\langle x, x\rangle) \subseteq x)$, *and* compose *is relation composition, defined in Example 2.6.*
4. *Select the smallest relation from* B, $C = \mathsf{select}(q)(B)$, *where* $q(x) = \mathsf{forall}(\lambda y.x \subseteq y)(B)$.
5. *Finally, flatten out* C *to get* $\mathsf{tc}(r) = \mu(C)$.

Parity can be defined in the same spirit, and actually any fixpoint query can be defined in $\mathcal{NRA}(\mathsf{powerset})$. We leave this proof as an exercise for the reader. But the above algorithm for tc is exponential, because at step 2 we construct the powerset of $v \times v$, requiring 2^{n^2} space, where $n = \mathsf{card}(v)$. We prove in this paper that *any* way of expressing tc in $\mathcal{NRA}(\mathsf{powerset})$ requires exponential space.

2.3 Evaluation in $\mathcal{NRA}(\mathsf{powerset})$

So far we have defined:

1.
   - A semantic notion of complex object.
   - A syntactic notation for complex objects (Definition 2.1).
2.
   - A semantic notion of a query (a query is a function).

- Syntactic notations for queries, as expressions in the languages $\mathcal{NRA}$ and $\mathcal{NRA}(\mathsf{powerset})$.

At the semantic level, queries denote *only* functions. That is, for a complex object Γ and query φ, $\Gamma' = \varphi(\Gamma)$ is simply the value of φ at Γ. At the syntactic level, an expression f in $\mathcal{NRA}(\mathsf{powerset})$ denotes an *algorithm*. In addition to defining a function, an algorithm also says *how* that function has to be computed. We describe this algorithm by defining an *evaluation strategy* for these languages, which states for any expression f and any complex object C the sequence of steps to be performed in order to compute $f(C)$. Obviously, different expressions f, f' denoting the same function φ may perform different steps when applied to a complex object C.

Several evaluation strategies could be conceived for the language $\mathcal{NRA}(\mathsf{powerset})$. The one we pick is the *call-by-value*, or *eager* evaluation strategy. Essentially it says that, in order to compute $f(e)$ one has to:

1. fully evaluate the subexpression e, and
2. apply f to the result.

It could be also called a "naive" evaluation strategy, since it does not do any optimizations for computing the query. The main result of the paper only holds for *this* evaluation strategy, and may fail for other evaluation strategies, see Section 8.

Instead of describing the evaluation strategy for $\mathcal{NRA}(\mathsf{powerset})$ in a step-by-step manner, we adopt the natural semantics style [Kah87]. Under this style, for some function expression $f \in \mathcal{NRA}(\mathsf{powerset})$ and complex objects C, C', we write $f(C) \Downarrow C'$ to mean "$f(C)$ evaluates to C'". The set of rules defining the binary relation $\Downarrow$ are given in Figure 3.

An evaluation $f(C) \Downarrow C'$ (which we sometimes abbreviate $f(C) \Downarrow$), can be viewed as a tree, called *derivation tree*, whose nodes are labeled by the rules above, and whose root contains a rule with $f(C) \Downarrow C'$ as its conclusion. If there is a node in the derivation tree labeled by a rule with $g(C'') \Downarrow C'''$ in the conclusion, then we say that C'', C''' both *occur* in the derivation tree. For a given expression f, the *height* of the evaluation tree $f(C) \Downarrow C'$ is bounded by a constant depending only on f, not on the complex objects. But the width of this tree may depend on C, because the branching factor at each node may depend on the size of the complex object(s) at that node (see the map rule).

Example 2.11 *Let unnest* $\stackrel{\text{def}}{=} \mu \circ \mathsf{map}(\Delta_{\mathsf{left}})$. *The derivation tree of:*

$$\mu \circ \mathsf{map}(\Delta_{\mathsf{left}})(\{\langle a, \{x\}\rangle, \langle b, \{y, z\}\rangle\}) \Downarrow \{\langle a, x\rangle, \langle b, y\rangle, \langle b, z\rangle\}$$

is given in Figure 4.

Although a ternary relation, $\Downarrow$ in fact denotes a function, as proved by the following Proposition:

Proposition 2.12 *The relation $\Downarrow$ is* **deterministic** *and* **total**, *i.e.* $\forall f \in \mathcal{NRA}(\mathsf{powerset})$ *and for any complex object C, there is exactly one complex object C' such that $f(C) \Downarrow C'$. Therefore we may talk about the evaluation $f(C) \Downarrow$, instead of $f(C) \Downarrow C'$.*

The proof is done by induction on the structure of f and is omitted.

$$\frac{}{\mathsf{id}(C) \Downarrow C} \qquad \frac{f(C_1) \Downarrow C_2 \qquad g(C_2) \Downarrow C_3}{(g \circ f)(C_1) \Downarrow C_3}$$

$$\frac{}{!\ C \Downarrow \langle\rangle} \qquad \frac{f_1(C) \Downarrow C_1 \quad f_2(C) \Downarrow C_2}{\langle f_1, f_2\rangle(C) \Downarrow \langle C_1, C_2\rangle}$$

$$\frac{}{\pi_1(\langle C_1, C_2\rangle) \Downarrow C_1} \qquad \frac{}{\pi_2(\langle C_1, C_2\rangle) \Downarrow C_2}$$

$$\frac{f(C_1) \Downarrow C_1' \quad \ldots \quad f(C_n) \Downarrow C_n'}{\mathsf{map}(f)\{C_1, \ldots, C_n\} \Downarrow \{C_1'\} \cup \ldots \cup \{C_n'\}}$$

$$\frac{}{\eta(C) \Downarrow \{C\}} \qquad \frac{}{\mu\{C_1, \ldots, C_n\} \Downarrow C_1 \cup \ldots \cup C_n}$$

$$\frac{}{\Delta_{\mathsf{left}}(\langle C, \{C_1, \ldots, C_n\}\rangle) \Downarrow \{\langle C, C_1\rangle, \ldots, \langle C, C_n\rangle\}}$$

$$\frac{}{\emptyset(\langle\rangle) \Downarrow \{\}} \qquad \frac{}{\cup(\langle C_1, C_2\rangle) \Downarrow C_1 \cup C_2}$$

$$\frac{}{= (\langle x, x\rangle) \Downarrow \mathsf{T}} \qquad \frac{}{= (\langle x, y\rangle) \Downarrow \mathsf{F}} \text{ when } x \neq y.$$

$$\frac{}{\mathsf{empty}(\{\}) \Downarrow \mathsf{T}} \qquad \frac{}{\mathsf{empty}(\{C_1, \ldots\}) \Downarrow \mathsf{F}}$$

$$\frac{}{\mathsf{true}(\langle\rangle) \Downarrow \mathsf{T}} \qquad \frac{}{\mathsf{false}(\langle\rangle) \Downarrow \mathsf{F}}$$

$$\frac{f(C_1) \Downarrow \mathsf{T} \qquad f_1(C_1) \Downarrow C_2}{\mathsf{if}\ f\ \mathsf{then}\ f_1\ \mathsf{else}\ f_2(C_1) \Downarrow C_2} \qquad \frac{f(C_1) \Downarrow \mathsf{F} \qquad f_2(C_1) \Downarrow C_2}{\mathsf{if}\ f\ \mathsf{then}\ f_1\ \mathsf{else}\ f_2(C_1) \Downarrow C_2}$$

$$\frac{}{\mathsf{powerset}(\{C_1, \ldots, C_n\}) \Downarrow \{C_1', \ldots, C_{2^n}'\}}$$
where $C_1', \ldots, C_{2^n}'$ are the subsets of $\{C_1, \ldots, C_n\}$

Figure 3: The definition of the evaluation relation $f(C) \Downarrow C'$.

$$\frac{}{\Delta_{left}(< a, \{x\} >) \Downarrow \{< a, x >\}}$$

$$\frac{}{\Delta_{left}(< b, \{y, z\} >) \Downarrow \{< b, y >, < b, z >\}}$$

$$\frac{\Delta_{left}(< a, \{x\} >) \Downarrow \{< a, x >\} \qquad \Delta_{left}(< b, \{y, z\} >) \Downarrow \{< b, y >, < b, z >\}}{\mathsf{map}(\Delta_{left})(\{< a, \{x\} >, < b, \{y, z\} >\}) \Downarrow \{\{< a, x >\}, \{< b, y >, < b, z >\}\}}$$

$$\frac{}{\mu(\{\{< a, x >\}, \{< b, y >, < b, z >\}\}) \Downarrow \{< a, x >, < b, y >, < b, z >\}}$$

$$\frac{\begin{array}{c}\mathsf{map}(\Delta_{left})(\{< a, \{x\} >, < b, \{y, z\} >\}) \Downarrow \{\{< a, x >\}, \{< b, y >, < b, z >\}\} \\ \mu(\{\{< a, x >\}, \{< b, y >, < b, z >\}\}) \Downarrow \{< a, x >, < b, y >, < b, z >\}\end{array}}{\mu \circ \mathsf{map}(\Delta_{left})(\{< a, \{x\} >, < b, \{y, z\} >\}) \Downarrow \{< a, x >, < b, y >, < b, z >\}}$$

Figure 4: An example of a derivation tree.

Definition 2.13 *The* **complexity** $\mathsf{complex}(f,C)$ *of the evaluation* $f(C)\Downarrow$ *is defined to be the size of the largest complex object occurring in the derivation tree of* $f(C)\Downarrow$*:*

$$\mathsf{complex}(f,C) \stackrel{\text{def}}{=} \max\{\mathsf{size}(C') \mid C' \textit{ occurs in the derivation tree } f(C)\Downarrow\}$$

This notion gives a reasonable account on the amount of space necessary to compute $f(C)\Downarrow$. Indeed, consider another reasonable candidate for a complexity measure of $f(C)\Downarrow$, namely *the total number of nodes* of the evaluation tree. This is bounded by $\mathsf{complex}(f,C)^h$, where h is an integer depending only on f, giving an upper bound on the height of the derivation tree of $f(C)\Downarrow$. Yet another alternative, *the sum of the sizes of all complex objects* occurring in the derivation tree of $f(C)\Downarrow$, is bounded by $\mathsf{complex}(f,C)^{h+1}$.

3 Main Results

The main result of this paper consists in proving that $\mathcal{NRA}(\mathsf{powerset})$, when equipped with the evaluation strategy described in Subsection 2.3, needs exponential space in order to compute transitive closure. More precisely we prove that for a particular sequence of binary relations $r_0, r_1, r_2, \ldots, r_n, \ldots$, if some expression f in $\mathcal{NRA}(\mathsf{powerset})$ computes the transitive closure of every r_n, then $\mathsf{complex}(f, r_n) \geq 2^{cn}$, for some constant $c > 0$.

We take $r_n \stackrel{\text{def}}{=} \{\langle 0,1\rangle, \langle 1,2\rangle, \langle 2,3\rangle, \ldots, \langle n-1,n\rangle\}$, i.e. r_n is a complex object of type $\{\mathbb{Z}\times\mathbb{Z}\}$ representing a particular binary relation (a chain of length n). Furthermore, let $q_n \stackrel{\text{def}}{=} \mathsf{tc}(r_n)$ be its transitive closure, that is

$$q_n = \{\langle x,y\rangle \mid 0 \leq x < y \leq n\}$$

We state our main result below:

Theorem 3.1 *For any expression* $f \in \mathcal{NRA}(\mathsf{powerset})$ *of type* $\{\mathbb{Z}\times\mathbb{Z}\} \to \{\mathbb{Z}\times\mathbb{Z}\}$ *such that* $f(r_n) \Downarrow q_n$ *for every* $n \geq 0$*, the complexity of* $f(r_n)\Downarrow$ *is* $\Omega(2^{cn})$*, for some* $c > 0$*.*

The proof is given in Section 7. As a consequence, transitive closure is not expressible in an efficient way in $\mathcal{NRA}(\mathsf{powerset})$.

For the proof of Theorem 3.1 we use a direct, brute force approach. Namely we introduce a notation for sequences of complex objects, indexed by n; we call this notation *abstract expressions* (see Definition 4.4). E.g. under this notation, r_n will be written as $\{\langle x-1, x\rangle \mid x = 0, \mathsf{n}; x \neq 0\}$: here n is part of the syntax for the abstract expressions, and its meaning is n. Nested relations can be expressed as abstract expressions too, like e.g. $\{\langle x, \{y \mid y = 0, \mathsf{n}; x \neq y\}\rangle \mid x = 0, \mathsf{n};\}$. The abstract expressions were carefully designed to enjoy the properties discussed below, which, together, prove our main result.

1. Abstract expressions are closed under applications of expressions in $\mathcal{NRA}$ (Theorem 5.2). However they are not closed under applications of functions in $\mathcal{NRA}(\mathsf{powerset})$. Indeed, any abstract expression denotes a sequence of complex objects whose sizes are bounded by $P(n)$, for some polynomial P (Proposition 6.3). In consequence, there is no abstract expression which can denote the result of, e.g., $\mathsf{powerset}(r_n)$.
2. The sequence r_n can be expressed as an abstract expressions, as suggested above (Fact 4.9).

3. But the transitive closures, $\mathsf{tc}(r_n)$, cannot be expressed by an abstract expression (Proposition 6.12). Certainly, the notation $\{\langle x, y\rangle \mid x, y = 0, \mathsf{n}; x < y\}$ stands for the transitive closure of r_n, but this is not an abstract expression, since in the definition of abstract expressions we only allow conditions of the form $x = y$ and $x \neq y$, and not conditions of the form $x < y$.

   To prove that $\mathsf{tc}(r_n)$ cannot be expressed with abstract expressions, we consider some abstract expressions of type $\{\mathbb{Z}^p\}$ of a special form; we call them *affine abstract expression.* Here $\{\mathbb{Z}^p\}$ for $p \geq 0$ is a shorthand for $\{\underbrace{\mathbb{Z} \times \ldots \times \mathbb{Z}}_{p \text{ times}}\}$. Any abstract expression of type $\{\mathbb{Z}^p\}$ is equivalent to a union of affine abstract expressions (Proposition 6.10). Moreover, the number of elements in a set denoted by an affine abstract expression is $n^k - O(n^{k-1})$, for some $k \geq 0$ (Proposition 6.3). We prove then in Proposition 6.12 that $\mathsf{tc}(r_n)$ cannot be expressed by an abstract expression since it has $n(n+1)/2 = n^2/2 - O(n)$ elements.

4. Any set expressed by an abstract expression, even with free variables, has either $O(1)$ or at least $\Omega(n)$ elements (this is implicitly stated in Proposition 7.13). For sets of type $\{\mathbb{Z}^p\}$, this follows from previously mentioned results, but for abstract expressions of arbitrary type we need a different proof. Technically, proving this property is the most difficult part of Theorem 3.1; Section 7 is devoted to it.

5. As a consequence, abstract expressions are closed under application of expressions in $\mathcal{NRA}(\mathsf{powerset})$ with polynomial complexity. Indeed, when applied to a set with $O(1)$ elements, powerset has polynomial complexity, and can be replaced with an expression in $\mathcal{NRA}$, hence item 1 above applies; when applied to a set with $\Omega(n)$ elements, its complexity is $\geq 2^{cn}$. In particular it follows that for every expression f in $\mathcal{NRA}(\mathsf{powerset})$, either (1) $\mathsf{complex}(f, r_n)$ is polynomial and then $f(r_n)$ can be expressed by some abstract expression, or (2) $\mathsf{complex}(f, r_n) \geq 2^{cn}$ for some $c > 0$ (Theorem 7.17).

Actually the proof of Theorem 3.1 implies more than that. Namely we will show that for any expression f of $\mathcal{NRA}(\mathsf{powerset})$, whose complexity on r_n is polynomial, all occurrences of powerset can be replaced by expressions in $\mathcal{NRA}$, such that the resulting expression f' (in $\mathcal{NRA}$) is equivalent to f on all r_n. Formally, for any number $m \geq 0$, define the m^{th} approximation of powerset to be $\mathsf{powerset}_m : \{t\} \rightarrow \{\{t\}\}$, s.t. $\forall x \in \{t\}$, $\mathsf{powerset}_m(x) = \{y \mid y \subseteq x, \mathsf{card}(y) \leq m\}$. Note that, for every $m \geq 0$, $\mathsf{powerset}_m$ is expressible in $\mathcal{NRA}$. For some expression $f \in \mathcal{NRA}(\mathsf{powerset})$, define f_m to be the m^{th} approximation of f, obtained by replacing all occurrences of powerset in f with $\mathsf{powerset}_m$. Then we can refine the statement of Theorem 3.1 to:

Proposition 3.2 *For any expression $f : \{\mathbb{Z} \times \mathbb{Z}\} \rightarrow t$ in $\mathcal{NRA}(\mathsf{powerset})$, either there exists some approximation f_m of f such that for $\forall n \geq 0$, $f_m(r_n) = f(r_n)$, or the complexity of $f(r_n) \Downarrow$ is $\Omega(2^{cn})$, for some $c > 0$.*

The proof is given in Section 7.

It was pointed out to us by Jan Van den Bussche [dB94] that the techniques developed for the proof of Theorem 3.1 and Proposition 3.2 apply also for parity (see Subsection 2.2), in the same way as for tc.

Proposition 3.3 *Any expression in $\mathcal{NRA}(\mathsf{powerset})$ for computing* parity *will have exponential complexity.*

The proof is given in Section 7.

We conjecture that Theorem 3.1 can be generalized, namely that any expression f in $\mathcal{NRA}(\mathsf{powerset})$ which has a polynomial complexity has an equivalent expression f' in $\mathcal{NRA}$. However in the general case the refinement in Proposition 3.2 of Theorem 3.1 no longer holds, suggesting that new proof techniques are needed for Theorem 3.1. Indeed, there exists functions f in $\mathcal{NRA}(\mathsf{powerset})$ which make "full use of $\mathsf{powerset}$", more precisely which are not equivalent to any f_m, $m \geq 0$. E.g. consider the following expression: $f : \{\mathbb{Z}\} \times \{\{\mathbb{Z}\}\} \to \mathbb{B}$, $f \stackrel{\text{def}}{=} (=_{\{\{\mathbb{Z}\}\}} \circ \langle \mathsf{powerset} \circ \pi_1, \pi_2 \rangle)$, with the meaning:

$$f(\langle x, y \rangle) = \begin{cases} \mathsf{T} & \text{when } \mathsf{powerset}(x) = y \\ \mathsf{F} & \text{otherwise} \end{cases}$$

That is, $f(\langle x, y \rangle)$ naively tests whether $\mathsf{powerset}(x) = y$, by actually computing $\mathsf{powerset}(x)$. Obviously, no approximation f_m of f will do the same job, but there exists another function f' in $\mathcal{NRA}$ equivalent to f, e.g. that which tests whether y contains the empty set, all singleton subsets of x, is closed under the union of two sets, and finally whether $\mu(y) \subseteq x$.

4 Abstract Expressions

4.1 Definitions and Basic Properties

Recall that $r_n = \{\langle 0, 1 \rangle, \langle 1, 2 \rangle, \ldots, \langle n-1, n \rangle\}$, for all $n \geq 0$. One can view r_n as a function from $\mathbb{N}$ to complex objects, assigning the complex object r_n to the number n. In the sequel we shall introduce a concise notation for functions from natural numbers to complex objects, inspired by the fact that we can express r_n as $\{\langle x-1, x \rangle \mid x = 0, \mathsf{n}; x \neq 0\}$. We call these notations *abstract expressions.* Assume an infinite set $\mathcal{X} = \{x, y, z, \ldots\}$ of variables to be given, and let n be a special symbol. Let $[n] \stackrel{\text{def}}{=} \{0, 1, 2, \ldots, n\}$.

Definition 4.1 *Let x stand for an arbitrary variable, c for an arbitrary integer. A* **simple expression** *has one of the following forms:*

1. $x + c$
2. c
3. $\mathsf{n} - c$

We abbreviate $x + 0$ to x, and $\mathsf{n} - 0$ to n, $x + (-i)$ to $x - i$, etc.

E.g. $7, -2, \mathsf{n} - 9, \mathsf{n}, x, x + 3, y - 8$ are simple expressions. But $x + y, \mathsf{n} - x, 2 * x$ are not. Note that the c in $\mathsf{n} - c$ may be a negative integer, e.g. $\mathsf{n} + 4$ is a legal simple expression. But we will mostly use expressions $\mathsf{n} - c$ with $c \geq 0$, hence our preference for denoting it like $\mathsf{n} - c$, instead of $\mathsf{n} + c$.

Definition 4.2 *A* **simple condition** *is a condition of the form $e = e'$, or $e \neq e'$, where e, e' are simple expressions. A* **condition** *is obtained by combining simple conditions with $\vee$ (or), $\wedge$ (and), $\neg$ (not),* true *and* false.

E.g. $(x = y + 5 \wedge y \neq 3) \vee (x \neq \mathsf{n} - 7 \wedge 3 \neq \mathsf{n} - 2)$ is a condition, however $x + 5 \leq y$ is not a condition, nor is $\mathsf{n} - x = 5$.

Definition 4.3 *In the sequel we will use the following abbreviations, with c a nonnegative integer:*

$$\begin{array}{rll} c \geq 0 & \textit{for} & \mathsf{true} \\ c \leq \mathsf{n} & \textit{for} & 0 \neq \mathsf{n} \wedge 1 \neq \mathsf{n} \wedge \ldots \wedge c-1 \neq \mathsf{n} \\ \mathsf{n}-c \geq 0 & \textit{for} & 0 \neq \mathsf{n} \wedge 1 \neq \mathsf{n} \wedge \ldots \wedge c-1 \neq \mathsf{n} \\ \mathsf{n}-c \leq \mathsf{n} & \textit{for} & \mathsf{true} \\ x-c \geq 0 & \textit{for} & x \neq 0 \wedge x \neq 1 \wedge \ldots \wedge x \neq c-1 \\ x-c \leq \mathsf{n} & \textit{for} & \mathsf{true} \\ x+c \geq 0 & \textit{for} & \mathsf{true} \\ x+c \leq \mathsf{n} & \textit{for} & x \neq \mathsf{n}-c+1 \wedge \ldots \wedge x \neq \mathsf{n}-1 \wedge x \neq \mathsf{n} \end{array}$$

Below we present the syntax for abstract expressions. As in the case of complex objects, abstract expressions are *typed*.

Definition 4.4

- $\langle\rangle$ *is an abstract expression of type unit.*
- *Any simple expression e is an abstract expression of type* $\mathbb{Z}$.
- $\mathsf{true}, \mathsf{false}$ *are abstract expressions of type* $\mathbb{B}$.
- *If A is an abstract expression of type t, $x_1, \ldots, x_k$ are k distinct variables, and C is a condition, then $\{A \mid x_1, \ldots, x_k = 0, \mathsf{n}; C\}$ is an abstract expression of type $\{t\}$. If $C = \mathsf{true}$, we write $\{A \mid x_1, \ldots, x_k = 0, \mathsf{n}\}$. If additionally $k = 0$, then we write $\{A\}$. If $C = \mathsf{false}$, we abbreviate it to $\emptyset$.*
- *If A_1, A_2 are abstract expression of type $\{t\}$, then $A_1 \cup A_2$ is also an abstract expression of type $\{t\}$.*
- *If A_1, A_2 are abstract expressions of type t and C is a condition, then* $\mathsf{if}\ C\ \mathsf{then}\ A_1\ \mathsf{else}\ A_2$ *is an abstract expression of type t.*
- *If A_1, A_2 are abstract expressions of types t_1 and t_2 respectively, then $\langle A_1, A_2\rangle$ is an abstract expression of type $t_1 \times t_2$.*

Examples of abstract expressions are: 3, $\mathsf{n}-5$, $\{\langle x, x+2\rangle \mid x = 0, \mathsf{n}; (x \neq \mathsf{n} \wedge x \neq \mathsf{n}-1)\}$. Also $\{2, 5, 12\}$ is an abbreviation for the abstract expression $\{2\} \cup \{5\} \cup \{12\}$.

As usual, we distinguish **bound** and **free** variables in some abstract expression A. E.g. in $\{\langle x, y\rangle \mid x = 0, \mathsf{n}; x \neq y+3\}$ x is bound and y is free. We adopt the convention that all bound variables are distinct and distinct from the free variables: this can always be obtained by renaming of the bound variables. A **closed abstract expression** is an abstract expression without free variables.

Recall that $\mathcal{X}$ is the set of all variables and $[n] = \{0, 1, \ldots, n\}$.

Definition 4.5 *Given $n \in \mathbb{N}$, an* **environment** *for n is a function $\rho : \mathcal{X} \to [n]$. We denote by $\mathsf{Env}^{[n]}$ the set of environments for n.*

Definition 4.6 *Let $\rho \in \mathsf{Env}^{[n]}$. The meaning of a simple expression e under ρ is denoted by $[\![e]\!]^{[n]}(\rho)$. The function $[\![e]\!]^{[n]} : \mathsf{Env}^{[n]} \to \mathbb{Z}$ is defined as follows:*

$$\begin{array}{rcl} [\![x + c]\!]^{[n]}(\rho) & \stackrel{\text{def}}{=} & \rho(x) + c \\ [\![c]\!]^{[n]}(\rho) & \stackrel{\text{def}}{=} & c \\ [\![\mathsf{n} - c]\!]^{[n]}(\rho) & \stackrel{\text{def}}{=} & n - c \end{array}$$

Definition 4.7 *The meaning of a condition C under ρ is denoted by $[\![C]\!]^{[n]}(\rho)$. The function $[\![C]\!]^{[n]} : \mathsf{Env}^{[n]} \to \mathbb{B}$ is defined as follows:*

$$\begin{array}{rcl} [\![e_1 = e_2]\!]^{[n]}(\rho) & \stackrel{\text{def}}{=} & \left\{ \begin{array}{ll} \mathsf{T} & \textit{if } [\![e_1]\!]^{[n]}(\rho) = [\![e_2]\!]^{[n]}(\rho) \\ \mathsf{F} & \textit{otherwise} \end{array} \right. \\ [\![e_1 \neq e_2]\!]^{[n]}(\rho) & \stackrel{\text{def}}{=} & \left\{ \begin{array}{ll} \mathsf{T} & \textit{if } [\![e_1]\!]^{[n]}(\rho) \neq [\![e_2]\!]^{[n]}(\rho) \\ \mathsf{F} & \textit{otherwise} \end{array} \right. \\ [\![C_1 \wedge C_2]\!]^{[n]}(\rho) & \stackrel{\text{def}}{=} & \left\{ \begin{array}{ll} \mathsf{T} & \textit{if } [\![C_1]\!]^{[n]}(\rho) = \mathsf{T} \textit{ and } [\![C_2]\!]^{[n]}(\rho) = \mathsf{T} \\ \mathsf{F} & \textit{otherwise} \end{array} \right. \\ [\![C_1 \vee C_2]\!]^{[n]}(\rho) & \stackrel{\text{def}}{=} & \left\{ \begin{array}{ll} \mathsf{T} & \textit{if } [\![C_1]\!]^{[n]}(\rho) = \mathsf{T} \textit{ or } [\![C_2]\!]^{[n]}(\rho) = \mathsf{T} \\ \mathsf{F} & \textit{otherwise} \end{array} \right. \\ [\![\neg C]\!]^{[n]}(\rho) & \stackrel{\text{def}}{=} & \left\{ \begin{array}{ll} \mathsf{T} & \textit{if } [\![C]\!]^{[n]}(\rho) = \mathsf{F} \\ \mathsf{F} & \textit{otherwise} \end{array} \right. \\ [\![\mathsf{true}]\!]^{[n]}(\rho) & \stackrel{\text{def}}{=} & \mathsf{T} \\ [\![\mathsf{false}]\!]^{[n]}(\rho) & \stackrel{\text{def}}{=} & \mathsf{F} \end{array}$$

Let $\rho \in \mathsf{Env}^{[n]}$, let $x_1, \ldots, x_k$ be distinct variables, and let $c_1, \ldots, c_k$ be k integers, $\forall i. 0 \leq c_i \leq n$. We write

$$\rho[x_1 = c_1, \ldots, x_k = c_k]$$

for the environment ρ' with:

$$\begin{array}{rcl} \rho'(x) & = & \rho(x), \text{ if } x \notin \{x_1, \ldots, x_k\} \\ \rho'(x_i) & = & c_i,\ 1 \leq i \leq k \end{array}$$

Definition 4.8 *Let A be an abstract expression of type t. The meaning of A under ρ is denoted by $[\![A]\!]^{[n]}(\rho)$. The partial function $[\![A]\!]^{[n]} : \mathsf{Env}^{[n]} \to t$ is defined as follows:*

- $[\![\langle\rangle]\!]^{[n]}(\rho) \stackrel{\text{def}}{=} \langle\rangle$.
- *For e a simple expression, $[\![e]\!]^{[n]}(\rho)$ is given in Definition 4.6.*
- *For $[\![\mathsf{true}]\!]^{[n]}(\rho)$ and $[\![\mathsf{false}]\!]^{[n]}(\rho)$ see Definition 4.7.*

- $[\![\langle A_1, A_2\rangle]\!]^{[n]}(\rho) \stackrel{\text{def}}{=} \langle [\![A_1]\!]^{[n]}(\rho), [\![A_2]\!]^{[n]}(\rho)\rangle$
- $[\![A_1 \cup A_2]\!]^{[n]}(\rho) \stackrel{\text{def}}{=} [\![A_1]\!]^{[n]}(\rho) \cup [\![A_2]\!]^{[n]}(\rho)$
- $[\![\mathsf{if}\ C\ \mathsf{then}\ A_1\ \mathsf{else}\ A_2]\!]^{[n]}(\rho) \stackrel{\text{def}}{=} \begin{cases} [\![A_1]\!]^{[n]}(\rho) & \textit{when}\ [\![C]\!]^{[n]}(\rho) = \mathsf{T} \\ [\![A_2]\!]^{[n]}(\rho) & \textit{when}\ [\![C]\!]^{[n]}(\rho) = \mathsf{F} \end{cases}$
- $[\![\{A \mid x_1, \ldots, x_k = 0, \mathsf{n}; C\}]\!]^{[n]}(\rho) \stackrel{\text{def}}{=}$
  $\{[\![A]\!]^{[n]}(\rho[x_1 = c_1, \ldots, x_k = c_k]) \mid 0 \leq c_1, \ldots, c_k \leq n, [\![C]\!]^{[n]}(\rho[x_1 = c_1, \ldots, x_k = c_k]) = \mathsf{T}\}$

Intuitively, in order to give a meaning to an abstract expression A, one has to (1) choose some n, and (2) assign values in $[n]$ to the free variables of A. Obviously $[\![A]\!]^{[n]}(\rho)$ will only depend on that part of ρ which assigns values to the variables which occur freely in A, that is, whenever ρ and ρ' agree on the free variables in A, $[\![A]\!]^{[n]}(\rho) = [\![A]\!]^{[n]}(\rho')$. Hence, the meaning $[\![A]\!]^{[n]}(\rho)$ of a closed abstract expression A is independent of ρ, and we denote it with $[\![A]\!]^{[n]}$.

Note that in Definition 4.6 till 4.8 we considered n to be fixed. Clearly the meaning depends on the value of n. We have, for instance:

n	$[\![\{\langle x-1, x+1\rangle \mid x = 0, \mathsf{n}; x \neq \mathsf{n}\}]\!]^{[n]}$
0	$\emptyset$
1	$\{\langle -1, 1\rangle\}$
2	$\{\langle -1, 1\rangle, \langle 0, 2\rangle\}$
3	$\{\langle -1, 1\rangle, \langle 0, 2\rangle, \langle 1, 3\rangle\}$

Fact 4.9 $[\![\{\langle x-1, x\rangle \mid x = 0, \mathsf{n}; x \neq 0\}]\!]^{[n]} = r_n$.

The following Lemma proves that the meaning of an abstract expression has a polynomial size.

Lemma 4.10 *For any abstract expression A there is some polynomial P such that $\forall n, \forall \rho \in \mathsf{Env}^{[n]}$:*

$$\mathsf{size}([\![A]\!]^{[n]}(\rho)) \leq P(n)$$

Proof. The proof follows by induction from the remarks below:

- $\mathsf{size}([\![\langle\rangle]\!]^{[n]}(\rho)) = 1$.
- $\mathsf{size}([\![e]\!]^{[n]}(\rho)) = 1$, where e is a simple expression.
- $\mathsf{size}([\![\mathsf{true}]\!]^{[n]}(\rho)) = \mathsf{size}([\![\mathsf{false}]\!]^{[n]}(\rho)) = 1$.
- $\mathsf{size}([\![\langle A_1, A_2\rangle]\!]^{[n]}(\rho)) = 1 + \mathsf{size}([\![A_1]\!]^{[n]}(\rho)) + \mathsf{size}([\![A_2]\!]^{[n]}(\rho))$.
- $\mathsf{size}([\![\mathsf{if}\ C\ \mathsf{then}\ A_1\ \mathsf{else}\ A_2]\!]^{[n]}(\rho)) \leq \max(\mathsf{size}([\![A_1]\!]^{[n]}(\rho)), \mathsf{size}([\![A_2]\!]^{[n]}(\rho)))$.

- $\mathsf{size}([\![\{A \mid x_1, \ldots, x_k = 0, \mathsf{n}; C\}]\!]^{[n]}(\rho)) \leq n^k \mathsf{size}([\![A]\!]^{[n]}(\rho'))$, for some ρ' of the form $\rho' = \rho[x_1 = c_1, \ldots, x_k = c_k]$.

$\square$

In fact one can observe that $\mathsf{size}([\![A]\!]^{[n]}(\rho)) = O(n^k)$, where k is the number of bound variables in A.

5 Abstract Expressions are Closed for $\mathcal{NRA}$

In this Section we prove that the set of abstract expressions is closed for the operations of $\mathcal{NRA}$. First we need to observe that the conditions introduced in Definition 4.2 enjoy a *quantifier elimination property* [BM75], i.e. they are equally expressive when extended with quantifiers. We will need two slightly different formulations of the quantifier elimination property (Proposition 5.1 below): the first for Theorem 5.2, and the second for Proposition 7.4.

Let C be a condition and x a variable. We say that another condition C', which does not have x as a free variable, **is equivalent to** $\exists x \in [\mathsf{n}].C$ iff $\forall n \geq 0$ the following condition holds:

$$[\![C']\!]^{[n]}(\rho) = \begin{cases} \mathsf{F} & \text{when } [\![C]\!]^{[n]}(\rho[x = k]) = \mathsf{F} \text{ for all } k \in [n] \\ \mathsf{T} & \text{when } [\![C]\!]^{[n]}(\rho[x = k]) = \mathsf{T} \text{ for some } k \in [n] \end{cases}$$

We say that C' **is equivalent to** $\exists x \in [\mathsf{n}].C$ **for** n **large enough**, if there exists $n_0 \geq 0$ such that $\forall n \geq n_0$ the above conditions holds.

Proposition 5.1 (Quantifier elimination) *Let C be some condition and x a variable. Then:*

1. *There exists a condition C' equivalent to $\exists x \in [\mathsf{n}].C$.*
2. *If C is a conjunction of negative simple conditions (i.e. of the form $e \neq e'$), let C' be the conjunction of only those conditions not involving x; the C' is equivalent to $\exists x \in [\mathsf{n}].C$ for n large enough.*

Proof. (See also [BM75], pp 49) Since $\exists x \in [\mathsf{n}].(C_1 \vee \ldots \vee C_k)$ is equivalent to $(\exists x \in [\mathsf{n}].C_1) \vee \ldots \vee (\exists x \in [\mathsf{n}].C_k)$, for item 1 it suffices to consider the case when C is a conjunction of simple conditions, as for item 2. Essentially C asserts some *positive* conditions on x, some *negative* conditions on x, and some other conditions not involving x. Without loss of generality, we shall assume that the positive conditions on x are $x = e_1 \wedge \ldots \wedge x = e_p$, and the negative conditions are $x \neq e'_1 \wedge \ldots \wedge x \neq e'_q$, where $e_1, \ldots, e_p, e'_1, \ldots, e'_q$ do not mention x: indeed, simply replace more general conditions like $x + 2 = y + 6$ with $x = y + 4$, etc; also, discard conditions like $x = x$ and $x \not= x + 7$; finally, if any condition of the form $x \neq x$ or $x = x + 5$ occurs, then take C' to be false. When $p \neq 0$, then $\exists x \in [\mathsf{n}].C$ simply asserts that $e_1, \ldots, e_p$ are equal, between 0 and n, and that they are distinct from $e'_1, \ldots, e'_q$, so to obtain C' we replace in C all conditions on x with $e_1 = e_2 \wedge e_1 = e_3 \wedge \ldots e_1 = e_p \wedge e_1 \geq 0 \wedge e_1 \leq \mathsf{n} \wedge e_1 \neq e'_1 \wedge \ldots \wedge e_1 \neq e'_q$. (See Definition 4.3 for the abbreviations $e_1 \geq 0$ and $e_1 \leq \mathsf{n}$.) This proves item 1 for the case $p > 0$. When $p = 0$, i.e. there are no positive assertions on x, then $\exists x \in [\mathsf{n}].C$ essentially says that there is at least one value $x \in [n]$ distinct from $e'_1, \ldots, e'_q$. This is obviously true whenever $n > q$, hence by taking $n_0 = q + 1$ we prove item 2. To prove item 1 for $p = 0$, observe that when $n \leq q$, then

$\exists x \in [\mathsf{n}].C$ is true iff for some value $k \in [n]$ we have $e'_1 \neq k, \ldots, e'_q \neq k$. Hence, we construct C' by replacing in C the conditions on x with:

$$(q \leq \mathsf{n}) \vee \bigvee_{k=0,q} (k \leq \mathsf{n}) \wedge (e'_1 \neq k \ldots, e'_q \neq k)$$

□

We may apply repeatedly Proposition 5.1 to eliminate a sequence of existential quantifiers. E.g. item 2 implies that whenever C is a conjunction of negative simple conditions and C' is the conjunction of only those conditions not involving any of $x_1, \ldots, x_k$, then $\exists x_1 \in [\mathsf{n}]. \ldots \exists x_k \in [\mathsf{n}].C$ is equivalent to C' for n large enough.

Theorem 5.2 (Evaluation Theorem) *Let A be some (not necessarily closed) abstract expression of type t_1, and f an expression in $\mathcal{NRA}$ of type $t_1 \to t_2$. Then there is some abstract expression A' of type t_2 such that $f(A) \Downarrow A'$, meaning that $\forall n \geq 0$, $\forall \rho \in \mathsf{Env}^{[n]}$, $f([\![A]\!]^{[n]}(\rho)) \Downarrow [\![A']\!]^{[n]}(\rho)$.*

Proof. We prove it straightforwardly, by induction on the structure of f, and on the structure of A. Note first that if the type t_1 is a set $\{t'_1\}$, then A has one of the forms: $A_1 \cup A_2$, or $\mathsf{if}\ C\ \mathsf{then}\ A_1\ \mathsf{else}\ A_2$, or $\{A_3 \mid x_1, \ldots, x_k = 0, \mathsf{n}; C\}$, where A_1, A_2 are again abstract expressions of a set type.

Case id : $\mathsf{id}(A) \Downarrow A$.

Case $g \circ f$: If by induction $f(A) \Downarrow A'$ and $g(A') \Downarrow A''$, then $(g \circ f)(A) \Downarrow A''$.

Case $!$: $!(A) \Downarrow \langle\rangle$.

Case $\langle f, g\rangle(A)$: If by induction $f(A) \Downarrow A'$ and $g(A) \Downarrow A''$, then $\langle f, g\rangle(A) \Downarrow \langle A', A''\rangle$.

Case π_1, π_2 : Note that A can be either a pair $\langle A_1, A_2\rangle$, or a conditional expression $\mathsf{if}\ C\ \mathsf{then}\ A_1\ \mathsf{else}\ A_2$. In the first case $\pi_1(\langle A_1, A_2\rangle) \Downarrow A_1$. For the second case, simply observe that $\pi_1(\mathsf{if}\ C\ \mathsf{then}\ A_1\ \mathsf{else}\ A_2) = \mathsf{if}\ C\ \mathsf{then}\ \pi_1(A_1)\ \mathsf{else}\ \pi_1(A_2)$.

Case $\mathsf{map}(f)$: If by induction we have $f(A) \Downarrow A'$, then clearly we have that $\mathsf{map}(f)(\{A \mid x_1, \ldots, x_k = 0, \mathsf{n}; C\}) \Downarrow \{A' \mid x_1, \ldots, x_k = 0, \mathsf{n}; C\}$. Furthermore:

$$\mathsf{map}(f)(\mathsf{if}\ C\ \mathsf{then}\ A_1\ \mathsf{else}\ A_2) = \mathsf{if}\ C\ \mathsf{then}\ \mathsf{map}(f)(A_1)\ \mathsf{else}\ \mathsf{map}(f)(A_2)$$

and $\mathsf{map}(f)(A_1 \cup A_2) = \mathsf{map}(f)(A_1) \cup \mathsf{map}(f)(A_2)$.

Case η : $\eta(A) \Downarrow \{A\}$.

Case μ : First we consider the case when $A = \{A' \mid x_1, \ldots, x_k = 0, \mathsf{n}; C\}$. Here we look at the structure of A':

- $\mu(\{A'_1 \cup A'_2 \mid x_1, \ldots, x_k = 0, \mathsf{n}; C\}) = \mu(\{A'_1 \mid x_1, \ldots, x_k = 0, \mathsf{n}; C\}) \cup \mu(\{A'_2 \mid x_1, \ldots, x_k = 0, \mathsf{n}; C\})$
- $\mu(\{\mathsf{if}\ C'\ \mathsf{then}\ A'_1\ \mathsf{else}\ A'_2 \mid x_1, \ldots, x_k = 0, \mathsf{n}; C\}) = \mu(\{A'_1 \mid x_1, \ldots, x_k = 0, \mathsf{n}; C \wedge C'\}) \cup \mu(\{A'_2 \mid x_1, \ldots, x_k = 0, \mathsf{n}; C \wedge \mathsf{not}\ C'\})$.
- $\mu(\{\{A'' \mid y_1, \ldots, y_l = 0, \mathsf{n}; C'\} \mid x_1, \ldots, x_k = 0, \mathsf{n}; C\}) \Downarrow \{A'' \mid x_1, \ldots, x_k, y_1, \ldots, y_l = 0, \mathsf{n}; C \wedge C'\}$.

Furthermore observe that $\mu(\mathsf{if}\ C\ \mathsf{then}\ A_1\ \mathsf{else}\ A_2) = \mathsf{if}\ C\ \mathsf{then}\ \mu(A_1)\ \mathsf{else}\ \mu(A_2)$ and $\mu(A_1 \cup A_2) = \mu(A_1) \cup \mu(A_2)$.

Case Δ_{left} :

$$\begin{array}{rcl}\Delta_{\mathsf{left}}(\langle A', \{A'' \mid x_1,\ldots,x_k = 0,\mathsf{n}; C\}\rangle) & \Downarrow & \{\langle A', A''\rangle \mid x_1,\ldots,x_k = 0,\mathsf{n}; C\} \\ \Delta_{\mathsf{left}}(\langle A, A' \cup A''\rangle) & = & \Delta_{\mathsf{left}}(\langle A, A'\rangle) \cup \Delta_{\mathsf{left}}(\langle A, A''\rangle) \\ \Delta_{\mathsf{left}}(\langle A, \mathsf{if}\ C\ \mathsf{then}\ A_1\ \mathsf{else}\ A_2\rangle) & = & \mathsf{if}\ C\ \mathsf{then}\ \Delta_{\mathsf{left}}(\langle A, A_1\rangle) \\ & & \mathsf{else}\ \Delta_{\mathsf{left}}(\langle A, A_2\rangle)\end{array}$$

Case $\emptyset$: $\emptyset(A) \Downarrow \emptyset$.

Case $\cup$: $\cup(\langle A_1, A_2\rangle) \Downarrow A_1 \cup A_2$, while $\cup(\mathsf{if}\ C\ \mathsf{then}\ A_1\ \mathsf{else}\ A_2) = \mathsf{if}\ C\ \mathsf{then}\ \cup(A_1)\ \mathsf{else}\ \cup(A_2)$.

Case $=$: Recall that $=$ is defined only on integers. Then $=(\langle e_1, e_2\rangle) \Downarrow \mathsf{if}\ (e_1 = e_2)\ \mathsf{then\ true\ else\ false}$. The other cases are trivial.

Case empty : Here we first prove that for any A there is some condition C such that $\forall n \geq 0$, $\forall \rho \in \mathsf{Env}^{[n]}$, $[\![\mathsf{empty}(A)]\!]^{[n]}(\rho) = [\![C]\!]^{[n]}(\rho)$. Then set $\mathsf{empty}(A) \Downarrow \mathsf{if}\ C\ \mathsf{then\ true\ else\ false}$. We prove the existence of such a C by induction on A. For $A = \{A' \mid x_1,\ldots,x_k = 0,\mathsf{n}; C\}$, $\mathsf{empty}(A)$ is equivalent to the negation of $\exists x_1 \in [\mathsf{n}] \ldots \exists x_k \in [\mathsf{n}].C$. Here we apply Proposition 5.1 item 1, to get $\mathsf{empty}(A)$ equivalent to some C'. For $A = A_1 \cup A_2$, observe that $\mathsf{empty}(A) = \mathsf{empty}(A_1) \wedge \mathsf{empty}(A_2)$. Finally, for $A = \mathsf{if}\ C\ \mathsf{then}\ A_1\ \mathsf{else}\ A_2$, $\mathsf{empty}(A) = (C \wedge \mathsf{empty}(A_1)) \vee (\mathsf{not}\ C \wedge \mathsf{empty}(A_2))$.

Case $\mathsf{true}, \mathsf{false}$: $\mathsf{true}(A) \Downarrow \mathsf{true}$, $\mathsf{false}(A) \Downarrow \mathsf{false}$.

Case if : To find A' such that $f(A) \Downarrow A'$ where $f = \mathsf{if}\ f_1\ \mathsf{then}\ f_2\ \mathsf{else}\ f_3$, start by computing $f_i(A) \Downarrow A_i$, $i = 1, 2, 3$. Then inspect the structure of A_1: because it is of type $\mathbb{B}$, it is true, false, or a tree of if's having true's and false's on the leaves. Then A' is obtained by replacing every occurrence of true with A_2 and every occurrence of false with A_3 in this tree. E.g. when $A_1 = \mathsf{if}\ C\ \mathsf{then}\ (\mathsf{if}\ C'\ \mathsf{then\ false\ else\ true})\ \mathsf{else\ false}$, then $A' = \mathsf{if}\ C\ \mathsf{then}\ (\mathsf{if}\ C'\ \mathsf{then}\ A_3\ \mathsf{else}\ A_2)\ \mathsf{else}\ A_3$.

□

The proof of Theorem 5.2, together with the need to express r_n, constitutes a justification for the choice of the syntax of abstract expressions. E.g. to handle the case for μ, we need more than one variable, and the condition C in the abstract expressions of the form $\{A \mid x_1,\ldots,x_k = 0,\mathsf{n}; C\}$. The case for $\cup$ explains the need for having $A_1 \cup A_2$ as a valid abstract expression. To handle empty and $=$ we need abstract expressions with if-conditionals.

Theorem 5.2 does not generalize in the same form to $\mathcal{NRA}(\mathsf{powerset})$, i.e. there are expressions f in the language $\mathcal{NRA}(\mathsf{powerset})$ and abstract expressions A such that for no A' do we have $f(A) \Downarrow A'$. Indeed, by Lemma 4.10, there is no abstract expression denoting $\mathsf{powerset}(\{x \mid x = 0,\mathsf{n}\})$, so it suffices to take $f = \mathsf{powerset}$ and $A = \{x \mid x = 0,\mathsf{n}\}$.

6 Abstract Expressions Cannot Express $\mathsf{tc}(r_n)$

For the rest of the paper we shall introduce some abbreviations. First we shall write $\mathbb{Z}^k$ for the product $\mathbb{Z} \times (\mathbb{Z} \times (\ldots(\mathbb{Z} \times \mathbb{Z})\ldots))$, and $\langle x_1,\ldots,x_k\rangle$ for $\langle x_1, \langle x_2, \langle \ldots, \langle x_{k-1}, x_k\rangle \ldots\rangle\rangle\rangle$. Moreover, we denote the latter with $\vec{x}$.

We are interested in the satisfiability of conditions for n large enough. Thus, we say that some condition $C(\vec{x})$ is **satisfiable** iff $\exists n_0 \geq 0$, such that $\forall n \geq n_0$, there is some environment $\rho \in \mathsf{Env}^{[n]}$ such that $[\![C]\!]^{[n]}(\rho) = \mathsf{true}$. E.g. the condition $x \neq 0 \wedge x \neq 1$ is satisfiable although for $n = 1$ there is no environment for n making this condition true, while the condition $x = 2 \wedge x = \mathsf{n} - 3$ is *not* satisfiable, because only for $n = 5$ can we find some x making it true.

We shall concentrate on **conjunctive conditions**, defined to be conjunctions of simple conditions.

Definition 6.1 *A closed abstract expression U of type $\{\mathbb{Z}^p\}$ is called an* **affine abstract expression** *iff it has the form:*

$$U = \{\langle A_1, \ldots, A_p \rangle \mid z_1, \ldots, z_k = 0, \mathsf{n}; C^-\}$$

with the following properties:

1. *Each A_i a simple expression, $i = 1, p$, such that each z_j is mentioned at least once in $\langle A_1, \ldots, A_p \rangle$.*
2. *C^- is a satisfiable conjunction of negative simple conditions.*
3. *For every $n \geq 0$, $[\![U]\!]^{[n]} \subseteq [n]^p$.*

We call $z_1, \ldots, z_k$ the **parameters** *of U, and say that U has k* **dimensions***, numbered $1, 2, \ldots, k$.*

The semantic condition 3 above requires that the simple abstract expressions A_i never overflow or underflow. In case it does not hold, but the other two conditions hold, one can easily enforce it by imposing the additional negative simple conditions $A_i \geq 0$ and $A_i \leq \mathsf{n}$ (see Definition 4.3 for these abbreviations). E.g. $U = \{\langle 2, z \rangle \mid z = 0, \mathsf{n}; \}$ would not qualify as an affine a.e. because for $n = 0$ and $n = 1$ the value 2 overflows, hence it violates item 3; however we may convert it to $U' = \{\langle 2, z \rangle \mid z = 0, \mathsf{n}; 0 \neq \mathsf{n} \wedge 1 \neq \mathsf{n}; \}$ which is an affine a.e. But note also that this trick does not always work, because it may lead to an unsatisfiable C^-, violating condition 2; e.g. $\{\langle \mathsf{n}{+}2, z \rangle \mid z = 0, \mathsf{n}; \}$ becomes, after adding the non overflow conditions, $\{\langle \mathsf{n}{+}2, z \rangle \mid z = 0, \mathsf{n}; \mathsf{n} \neq \mathsf{n} \wedge \mathsf{n} \neq \mathsf{n}{-}1\}$, which violates 2.

Example 6.2 *$U = \{\langle z_1, z_2 + 1, z_1 - 1 \rangle \mid z_1 \neq 0 \wedge z_2 \neq \mathsf{n} \wedge z_1 \neq z_2\}$ is an affine a.e. with $k = 2$ dimensions. Here $p = 3$.*

Our interest in affine abstract expressions is related to the fact that we can see at a glance how many elements the set $[\![U]\!]^{[n]}$ has: namely approximatively n^k, where k is the number of dimensions of U.

Proposition 6.3 *For any affine abstract expression U with k dimensions, $[\![U]\!]^{[n]}$ has $n^k - O(n^{k-1})$ elements. Hence, an affine a.e. with 0 dimensions has at most 1 element, and for any affine a.e. U, $[\![U]\!]^{[n]} \neq \emptyset$, for n sufficiently large.*

Proof. Consider some affine a.e. $U = \{\vec{A}(z) \mid \vec{z} = 0, \mathsf{n}; C^-\}$, where C^- is a conjunction of negative conditions $C^- = C_1^- \wedge \ldots \wedge C_m^-$. Then $[\![U]\!]^{[n]} = [\![\{\vec{A}(\vec{z}) \mid \vec{z} = 0, \mathsf{n}; \}]\!]^{[n]} - [\![U_1]\!]^{[n]} - \ldots - [\![U_m]\!]^{[n]}$, where $U_i = \{\vec{A}(\vec{z}) \mid \vec{z} = 0, \mathsf{n}; \mathsf{not}(C_i^-)\}$. Obviously $[\![U_i]\!]^{[n]}$ has $O(n^{k-1})$ elements, because $\mathsf{not}(C_i^-)$ is a positive simple condition on one or two of the $\vec{z}$'s. $\square$

Example 6.4 *The affine a.e. U of Example 6.2 has $n^2 - n + 1 = n^2 - O(n)$ elements, for $n \geq 2$. Here is some pathological case: take $V = \{2 \mid ; \mathsf{n} \neq 0 \wedge \mathsf{n} \neq 1\}$, an affine a.e. with 0 dimensions. Then, for $n = 0$ and $n = 1$, $[\![V]\!]^{[0]}$ and $[\![V]\!]^{[1]}$ are empty, but for $n \geq 2$, $[\![V]\!]^{[n]}$ has exactly one element. In general, it can happen that an affine a.e. denotes the empty set, but only for a finite number of n's.*

Note that we always have $k \leq p$ because each simple expression A_i may mention at most one parameter z_j, and each z_j has to be mentioned at least once.

Intuitively an affine abstract expression describes a subset of $[n]^p$ *explicitly*, via the parameters $z_1, \ldots, z_k$. Alternatively, we may describe subsets of $[n]^p$ *implicitly*, by a condition on its coordinates $x_1, \ldots, x_p$; we call such an abstract expression *dual-affine*:

Definition 6.5 *A* **dual-affine abstract expression** *is an abstract expression of type $\{\mathbb{Z}^p\}$ of the form:*

$$A = \{\langle x_1, \ldots, x_p \rangle \mid x_1, \ldots, x_p = 0, \mathsf{n}; C\}$$

with C a conjunctive, satisfiable condition.

Compared to the affine abstract expressions, the dual affine abstract expressions are constrained to have exactly the variables $x_1, \ldots, x_p$ in their header, rather than arbitrary simple expressions; in compensation they may use both positive and negative conditions in C. Nevertheless, the two forms of abstract expressions turn out to be equivalent:

Proposition 6.6 *For any affine abstract expression U there exists an equivalent dual-affine abstract expression A, i.e. $[\![U]\!]^{[n]} = [\![A]\!]^{[n]}$, $\forall n \geq 0$. Conversely, for any dual-affine expression A there exists an equivalent affine abstract expression U.*

Proof. (Sketch) Given an affine abstract expression $U = \{\langle A_1, \ldots, A_p \rangle \mid z_1, \ldots, z_k = 0, \mathsf{n}; C^-\}$, we introduce new variables $x_1, \ldots, x_p$, and construct the equalities $x_1 = A_1, \ldots, x_p = A_p$. First consider all those equalities $x_i = A_i$ in which A_i mentions some parameter variable, say z_j; rearrange each of them in the form $z_j = x_i + c_i$. In this way we express every parameter z_j as a function of one, or several x_i's; if there are more then one, we equate these expressions pairwise, and call $C_1(\vec{x})$ the conjunction of these conditions. In addition, we impose the conditions $x_i + c_i \geq 0$ and $x_i + c_i \leq \mathsf{n}$ (see Definition 4.3); call $C_2(\vec{x})$ the resulting conditions. Next we consider all conditions $x_i = A_i$ in which A_i is a constant c, or $\mathsf{n} - c$; call $C_3(\vec{x})$ the conjunction of these conditions. Finally we define $C_4(\vec{x})$ to the the condition $C^-(\vec{z})$ in which every variable z_j is replaced with its corresponding $x_i + c_i$ (choose arbitrarily one such simple abstract expression, if there are more for the same z_j). Obviously:

$$[\![U]\!]^{[n]} = \{\langle x_1, \ldots, x_k \rangle \mid C_1 \wedge C_2 \wedge C_3 \wedge C_4\}$$

See Example 6.7 below.

For the converse, let $A = \{\vec{x} \mid \vec{x} = 0, \mathsf{n}; C(\vec{x})\}$ be a dual-affine abstract expression, with $C(\vec{x}) = C^-(\vec{x}) \wedge C^+(\vec{x})$, where $C^-(\vec{x})$ contains all negative conditions of $C(\vec{x})$, while $C^+(\vec{x})$ contains all positive conditions. Consider the smallest equivalence relation $x_i \equiv x_j$ on the set of variables $\{x_1, \ldots, x_p\}$, for which, for every positive simple condition $x_i = x_j + c$ in $C^+(\vec{x})$, we have $x_i \equiv x_j$. That is, $x_i \equiv x_j$ whenever they are related, directly or indirectly, by $C^+(\vec{x})$. We call an equivalence class *bound*, iff it contains some variable x_i for which a condition

$x_i = c$ or $x_i = \mathsf{n} - c$ is present in $C(\vec{x})$; else we call the equivalence class *free*. Since $C(\vec{x})$ is satisfiable, there is essentially only one way we can express such a variable x_i. Then, for each variable x_i from a bound equivalence classes we define $A_i \stackrel{\text{def}}{=} c$, or $A_i \stackrel{\text{def}}{=} \mathsf{n} - c$. Next let k be the number of free equivalence classes, and number them with $1, 2, \ldots, k$. We will select one variable x_i from each free equivalence class α, and rename it z_α, where $\alpha = 1, \ldots, k$. These variables $z_1, \ldots, z_k$ will be the parameters of the affine abstract expression. Each variable x_i occuring in a free equivalence class α_i, can be expressed as a function of the parameter z_{α_i}, e.g. $x_i = z_{\alpha_i} + c_i$, because it is directly or indirectly related to z_{α_i} in $C^+(\vec{x})$; here we define $A_i \stackrel{\text{def}}{=} z_{\alpha_i} + c_i$. There is essentially only one way in which x_i can be expressed as a function of z_{α_i}, or else the condition $C(\vec{x})$ would not be satisfiable. Finally define U to be the affine abstract expression:

$$U \stackrel{\text{def}}{=} \{\langle A_1, \ldots, A_p \rangle \mid \bigwedge_{i=1,p} (0 \leq A_i \leq \mathsf{n}) \wedge C^-(A_1(\vec{z}), \ldots, A_p(\vec{z}))\}$$

See Example 6.8 below. □

Example 6.7 *Consider the affine a.e.*

$$U = \{\langle z_1 + 1, z_1 - 1, z_2, 1 \rangle \mid z_1, z_2 = 0, \mathsf{n}; z_1 \neq 0 \wedge z_1 \neq \mathsf{n} \wedge \mathsf{n} \neq 0 \wedge z_1 \neq z_2 + 5\}$$

To obtain an equivalent dual affine abstract expression, we start by renaming $x_1 := z_1 + 1, x_2 := z_1 - 1, x_3 := z_2, x_4 := 1$. We rewrite the first three as $z_1 = x_1 - 1, z_1 = x_2 + 1, z_2 = x_3$. Of the two expressions for z_1 we pick $z_1 := x_1 - 1$, and define $C_1(\vec{x}) \stackrel{\text{def}}{=} x_1 - 1 = x_2 + 1$. C_2 will be the "non-overflow" condition $x_1 - 1 \geq 0$, i.e. $C_2(\vec{x}) \stackrel{\text{def}}{=} x_1 \neq 0$. Next, $C_3(\vec{x}) \stackrel{\text{def}}{=} x_4 = 1$, and $C_4(\vec{x}) \stackrel{\text{def}}{=} x_1 - 1 \neq 0 \wedge x_1 - 1 \neq \mathsf{n} \wedge \mathsf{n} \neq 0 \wedge x_1 - 1 \neq x_3 + 5$. We get the equivalent dual-affine abstract expression

$$\begin{aligned} A = \{ \; & \langle x_1, x_2, x_3, x_4 \rangle \mid x_1, x_2, x_3, x_4 = 0, \mathsf{n}; \\ & x_1 - 1 = x_2 + 1 \wedge x_1 \neq 0 \wedge x_4 = 1 \wedge x_1 - 1 \neq 0 \wedge x_1 - 1 \neq \mathsf{n} \wedge \mathsf{n} \neq 0 \wedge x_1 - 1 \neq x_3 + 5\} \end{aligned}$$

Example 6.8 *Consider the dual affine abstract expression*

$$A \stackrel{\text{def}}{=} \{\langle x_1, x_2, x_3 \rangle \mid x_1, x_2, x_3 = 0, \mathsf{n}; (x_1 = x_2) \wedge (x_2 = x_3 + 4) \wedge x_3 \neq 9\}$$

Here $C^+(x_1, x_2, x_3) = (x_1 = x_2) \wedge (x_2 = x_3 + 4)$, and $C^-(x_1, x_2, x_3) = (x_3 \neq 9)$. We have only one equivalence class, namely $\{x_1, x_2, x_3\}$, and we choose x_1 as its representative, which we rename z. So A is equivalent to the affine abstract expression:

$$U \stackrel{\text{def}}{=} \{(z, z, z - 4) \mid z = 0, \mathsf{n}; z - 4 \neq 9 \wedge z \neq 0 \wedge z \neq 1 \wedge z \neq 2 \wedge z \neq 3\}$$

As a consequence we have the following Corollary:

Corollary 6.9 *For any two affine a.e.'s U_1, U_2 of the same type, either $U_1 \cap U_2 = \emptyset$ for n large enough, or there exists another affine a.e. U equivalent to $U_1 \cap U_2$.*

Proof. By Proposition 6.6 above the affine abstract expressions U_1 and U_2 are equivalent to dual-affine abstract expressions A_1, A_2, where $A_1 = \{\vec{x} \mid \vec{x} = 0, \mathsf{n}; C_1(\vec{x})\}$ and $A_2 = \{\vec{x} \mid \vec{x} = 0, \mathsf{n}; C_2(\vec{x})\}$. Their intersection $U_1 \cap U_2$ is equivalent to $\{\vec{x} \mid \vec{x} = 0, \mathsf{n}; C_1 \wedge C_2\}$ which is either a dual affine a.e. (when $C_1 \wedge C_2$ is satisfiable), or empty for n large enough (when $C_1 \wedge C_2$ is not satisfiable). □

Proposition 6.10 *For any closed abstract expression A of type $\{\mathbb{Z}^p\}$, $p \geq 0$, for which $[\![A]\!]^{[n]} \subseteq [n]^p, \forall n \geq 0$, there are some m affine a.e. $U_1, \ldots, U_m$, such that A is equivalent to $U_1 \cup \ldots \cup U_m$, for n sufficiently large.*

Proof. We prove this by induction on the structure of A. The case when $A = A_1 \cup A_2$ is trivial. Suppose $A = \mathsf{if}\ C\ \mathsf{then}\ A_1\ \mathsf{else}\ A_2$. Because C is closed, it must be either true or false for n sufficiently large (e.g. it may be something like $\mathsf{n} = 2 \vee \mathsf{n} = 4$, which is false for $n > 4$). Hence A is either equivalent to A_1, or equivalent to A_2, for n sufficiently large, and we apply induction hypothesis. Finally, suppose $A = \{A' \mid \vec{x} = 0, \mathsf{n}; C\}$. Here we first "normalize" A', which is of type $\mathbb{Z}^p$, by "pulling up" the if-expressions, i.e. by replacing each subexpression of the form:

$$\langle \mathsf{if}\ C'\ \mathsf{then}\ A_1\ \mathsf{else}\ A_2, A_3 \rangle$$

with:

$$\mathsf{if}\ C'\ \mathsf{then}\ \langle A_1, A_3 \rangle\ \mathsf{else}\ \langle A_2, A_3 \rangle$$

and similarly for the second component. Next we "normalize" A itself, by replacing

$$\{\mathsf{if}\ C'\ \mathsf{then}\ A'\ \mathsf{else}\ A'' \mid \vec{x} = 0, \mathsf{n}; C\}$$

with

$$\{A' \mid \vec{x} = 0, \mathsf{n}; C \wedge C'\} \cup \{A'' \mid \vec{x} = 0, \mathsf{n}; C \wedge \mathsf{not}\ C'\}$$

Finally we only have to consider the case when A' is a tuple of simple expressions. Here we write C in disjunctive normal form $C = C_1 \vee \ldots \vee C_m$, and we get $\{A' \mid \vec{x} = 0, \mathsf{n}; C_1 \vee \ldots \vee C_m\} = \{A' \mid \vec{x} = 0, \mathsf{n}; C_1\} \cup \ldots \cup \{A' \mid \vec{x} = 0, \mathsf{n}; C_m\}$. We are not done yet, because the conditions $C_1, \ldots, C_m$ are not necessarily negative. But it is easy to transform each of them into a dual affine a.e. using Proposition 6.6. Finally, each of the m sets can be easily converted to an equivalent dual affine abstract expression. $\square$

Example 6.11 *Consider the abstract expression $A = (\mathsf{if}\ n = 0 \vee \mathsf{n} = 1\ \mathsf{then}\ \{0\}\ \mathsf{else}\ \{1, 2\})$ of type $\{\mathbb{Z}\}$. Then, for $n \geq 2$, A is equivalent to $\{1 \mid; \mathsf{n} \neq 0\} \cup \{2 \mid; \mathsf{n} \neq 0 \wedge \mathsf{n} \neq 1\}$, both being affine a.e. with 0 dimensions. Note that we cannot express A as a union of affine abstract expressions for* every *n.*

Proposition 6.12 *No abstract expression can denote $\mathsf{tc}(r_n)$ for all $n \geq 0$.*

Proof. Let on the contrary $\mathsf{tc}(r_n)$ be the meaning of some a.e. By Proposition 6.10 it is the meaning of a union of affine a.e.'s $U_1 \cup \ldots \cup U_m$. If some U_i would have 2 or more dimensions, then:

$$\mathsf{card}(\mathsf{tc}(r_n)) \geq \mathsf{card}([\![U_i]\!]^{[n]}) \geq n^2 - cn$$

for some constant $c \geq 0$. But since $\mathsf{card}(\mathsf{tc}(r_n)) = \frac{n(n+1)}{2}$, this introduces a contradiction.

On the other hand, if all U_i would have at most 1 dimension, then:

$$\mathsf{card}(\mathsf{tc}(r_n)) \leq mn$$

which is again a contradiction, because m is a constant. $\square$

7 Powerset Applied to Abstract Expressions

We shall generalize Theorem 5.2 to expressions in $\mathcal{NRA}(\mathsf{powerset})$ in a more subtle way (see Theorem 7.17). Namely we show that for any expression f in $\mathcal{NRA}(\mathsf{powerset})$ and any abstract expression A, $f(A)$ is either some abstract expression, or $f(A) \Downarrow$ requires exponential space to compute. The key step will be to show that any intermediate result in the derivation tree of $f(A) \Downarrow$ that can be expressed by some abstract expression of a set type, say $\{A'(x) \mid x = 0, \mathsf{n}\}$, has either $O(1)$ or at least $\Omega(n)$ elements. But this will require a lengthly technical argument. To see the problem, suppose that A' has at most x as a free variable. At a first glance it seems that the set $\{A'(x) \mid x = 0, \mathsf{n}\}$ has $n+1$ elements, namely $A'(0), A'(1), \ldots, A'(n)$. However, some of these may be equal, e.g. when A' does not depend on x, then *all* are equal. So we need to count the number of distinct elements, and prove that this number is either $O(1)$ or at least $\Omega(n)$. For this, let x' be a fresh variable. By Theorem 5.2, the inequality $A'(x) \neq A'(x')$ can be expressed as a condition $C(x, x')$. Indeed, it suffices to compute $= (\langle A'(x), A'(x')\rangle) \Downarrow A''$, where $=$ is the equality expression defined in Example 2.5. Since A'' is of type $\mathbb{B}$, it must be essentially of the form $\mathsf{if}\ C(x, x')\ \mathsf{then\ false\ else\ true}$. Now $C(x, x')$ defines a binary relation, the "inequality relation" for A': whenever $C(x, x')$ is true, $A'(x)$ and $A'(x')$ are distinct elements of the set. The set $\{A' \mid x = 0, \mathsf{n};\}$ has at least m elements iff there is some sequence $s = [x^1, \ldots, x^m]$, $x^i \in [n]$, $i = 1, m$, such that $C(x^i, x^j)$ is true, for all i, j, $1 \leq i < j \leq m$. Borrowing from the graph-theoretic terminology, we will call such a sequence s *a clique*. Then the cardinality of $\{A' \mid x = 0, \mathsf{n};\}$ equals the size of the largest clique. Things become more complicated in the presence of free variables: if A has y as a free variable, i.e. $A(y) = \{A'(x, y) \mid x = 0, \mathsf{n}\}$, then for some values of y, the set A may have $O(1)$ elements, while for other values it may have $\Omega(n)$ elements.

Our plan for generalizing Theorem 5.2 to $\mathcal{NRA}(\mathsf{powerset})$ is the following:

- Generalize *affine abstract expressions* to *variable affine abstract expressions*. This is our technical tool for handling free variables.
- Prove that, if C is a conjunction of simple conditions, then its largest clique has size either $O(1)$ or $\Omega(n)$ (Proposition 7.13).
- Prove that for an arbitrary condition C, its largest clique has size either $O(1)$ or $\Omega(n)$ (Proposition 7.16).
- Apply these results to generalize Theorem 5.2 to $\mathcal{NRA}(\mathsf{powerset})$ (Theorem 7.17).

7.1 Variable Affine Abstract Expressions

Any dual-affine abstract expression $\{\vec{x} \mid \vec{x} = 0, \mathsf{n}; C(\vec{x})\}$ is, by Proposition 6.6, equivalent to some affine abstract expressions and, hence, by Proposition 6.3 will have $\Theta(n^p)$ elements. We need to look also at sets given by abstract expressions of the form $\{\vec{x} \mid \vec{x} = 0, \mathsf{n}; C(\vec{x}, \vec{y})\}$, which are parameterized by the variables $\vec{y}$. It turns out that such a set is equivalent to a *variable affine expressions* $V(\vec{y})$.

Definition 7.1 *A* **variable affine abstract expression** V *is an expression of type* $\{\mathbb{Z}^p\}$ *of the form*

$$V = \{\langle A_1(\vec{z}, \vec{y}), \ldots, A_p(\vec{z}, \vec{y})\rangle \mid \vec{z} = 0, \mathsf{n}; C^-(\vec{z}, \vec{y})\}$$

satisfying the following conditions:

1. $A_1, \ldots, A_p$ *are simple expressions,* $\vec{z}$ *is* $\langle z_1, \ldots, z_k \rangle$*, and every variable* z_j *occurs in* $\langle A_1, \ldots, A_p \rangle$.
2. C^- *is a satisfiable conjunction of negative simple conditions.*
3. *For every* $n \geq 0$ *and every* $\rho \in \mathsf{Env}^{[n]}$, $[\![V]\!]^{[n]}(\rho) \subseteq [n]^p$.

We write $V(\vec{y})$ *to emphasize its free variables* $\vec{y}$*, and say that* $V(\vec{y})$ *has* k **dimensions**.

Example 7.2 $V(y) = \{\langle \mathsf{n}, z+1, y-1 \rangle \mid z = 0, \mathsf{n}; z \neq \mathsf{n} \wedge z \neq y - 3 \wedge y \neq 0\}$ *is a variable affine a.e., with 1 dimension. Note that* $V(y)$ *is empty when* $y = 0$.

Definition 7.3 *We say that an affine a.e.* U *and a variable affine a.e.* $V(\vec{y})$ *are* **associated** *iff for* n *large enough* $\vec{y} \in U \Rightarrow V(\vec{y}) \neq \emptyset$*; more precisely, when* $\exists n_0 \geq 0$ *such that* $\forall n \geq n_0$, $\forall \rho \in \mathsf{Env}^{[n]}$, $[\![\vec{y}]\!]^{[n]}(\rho) \in [\![U]\!]^{[n]} \Longrightarrow [\![V(\vec{y})]\!]^{[n]}(\rho) \neq \emptyset$.

Proposition 7.4 *For any variable affine a.e.* $V(\vec{y})$ *there is exists an affine abstract expression* U *such that* $\exists n_0 \geq 0. \forall n \geq n_0. \forall \rho \in \mathsf{Env}^{[n]}. ([\![\vec{y}]\!]^{[n]}(\rho) \in [\![U]\!]^{[n]} \iff [\![V(\vec{y})]\!]^{[n]}(\rho) \neq \emptyset)$*. We call* U *the* **maximal** *affine abstract expression associated with* $V(\vec{y})$.

Proof. Let $V(\vec{y}) = \{\langle A_1, \ldots, A_p \rangle \mid \vec{z} = 0, \mathsf{n}; C(\vec{z}, \vec{y})\}$. Split $C(\vec{z}, \vec{y})$ into $C_1(\vec{y}) \wedge C_2(\vec{z}, \vec{y})$, where $C_1(\vec{y})$ contains all simple conditions mentioning only the variables $\vec{y}$, and every simple condition in $C_2(\vec{z}, \vec{y})$ mentions at least one variable from $\vec{z}$. Recall that both C_1 and C_2 are conjunctions of negative simple conditions. Let U be $\{\vec{y} \mid \vec{y} = 0, \mathsf{n}; C_1(\vec{y})\}$. Certainly $[\![V(\vec{y})]\!]^{[n]}(\rho) \neq \emptyset \Longrightarrow [\![\vec{y}]\!]^{[n]}(\rho) \in [\![U]\!]^{[n]}$, so U is "maximal". It remains to show that U and $V(\vec{y})$ are associated. Let n be large enough (to be specified later) and $\vec{y} \in U$. We want to show that $V(\vec{y}) \neq \emptyset$, or, equivalently, $\exists \vec{z} \in [\mathsf{n}]^k. C(\vec{z}, \vec{y})$ By item 2 of Proposition 5.1 we can choose n large enough such that $\exists \vec{z} \in [\mathsf{n}]^k. C(\vec{z}, \vec{y})$ is equivalent to the conjunction of only those conditions in $C(\vec{z}, \vec{y})$ which do not mention $\vec{z}$, i.e. to $C_1(\vec{y})$. The latter is true, because $\vec{y} \in U$. □

This "maximal" U associated with $V(\vec{y})$ captures the negative conditions on $\vec{y}$ present in $V(\vec{y})$. Note that U will always have a maximal number of dimensions, i.e. q, the number of variables in $\vec{y}$. This result cannot be extended to arbitrary n's. E.g. let $V(y) = \{z \mid z = 0, \mathsf{n}; z \neq 0 \wedge z \neq y\}$. Then the maximal affine a.e. associated to $V(\vec{y})$ is $U = \{y \mid y = 0, \mathsf{n};\}$ and $n_0 = 2$. For $n = 1$, we have $1 \in U$, but $V(1) = \emptyset$.

The properties mentioned in Proposition 6.6 and Corollary 6.9 extend to variable affine abstract expressions. For that, we define a **variable dual-affine abstract expression** to be an abstract expression of type $\{\mathbb{Z}^p\}$ of the form $A = \{\vec{x} \mid \vec{x} = 0, \mathsf{n}; C(\vec{x}, \vec{y})\}$, with $C(\vec{x}, \vec{y})$ a satisfiable conjunction of simple conditions. Again we write $A(\vec{y})$ to emphasize the dependence on $\vec{y}$.

Proposition 7.5 *For any variable dual-affine abstract expression* $A(\vec{y}) = \{\vec{x} \mid \vec{x} = 0, \mathsf{n}; C(\vec{x}, \vec{y})\}$ *there exists an associated pair of affine abstract expressions* $U, V(\vec{y})$ *which are* **equivalent** *to* A*, i.e. such that*

$$\forall x. (\vec{x} \in A(\vec{y}) \iff \vec{y} \in U \wedge \vec{x} \in V(\vec{y}))$$

More precisely

$$\forall n \geq 0, \forall \rho \in \mathsf{Env}^{[n]}, \forall \vec{x} \in [n]^p. (\vec{x} \in [\![A(\vec{y})]\!]^{[n]}(\rho) \iff [\![\vec{y}]\!]^{[n]}(\rho) \in [\![U]\!]^{[n]} \wedge \vec{x} \in [\![V(\vec{y})]\!]^{[n]}(\rho))$$

Conversely, for any associated pair of affine abstract expression $U, V(\vec{y})$*, there exists an equivalent variable dual-affine abstract expression.*

Proof. (Sketch) For some variable dual-affine abstract expression $A(\vec{y}) = \{\vec{x} \mid \vec{x} = 0, \mathsf{n}; C(\vec{x}, \vec{y})\}$, we will construct an affine abstract expression U and a variable affine abstract expression $V(\vec{y}) = \{\vec{A}(\vec{z}, \vec{y}) \mid \vec{z} = 0, \mathsf{n}; C^-(\vec{y}, \vec{z})\}$ as follows. We proceed as in the proof of Proposition 6.6. First we separate $C(\vec{x}, \vec{y})$ into $C_1(\vec{y}) \wedge C_2(\vec{x}, \vec{y})$, where $C_2(\vec{x}, \vec{y})$ contains only conditions mentioning at least one x. As in the proof of Proposition 6.6, we split $C_2(\vec{x}, \vec{y})$ into *positive* and *negative conditions*, $C_2(\vec{x}, \vec{y}) = C_2^+(\vec{x}, \vec{y}) \wedge C_2^-(\vec{x}, \vec{y})$, and define the equivalence relation $x_i \equiv x_j$ on the variables $x_1, \ldots, x_p$, stating that x_i and x_j are related in $C_2^+(\vec{x}, \vec{y})$. As in Proposition 6.6, we call an equivalence class *bound*, iff it contains a variable x_i for which a condition $x_i = c$, or $x_i = \mathsf{n} - c$, or $x_i = y_j + c$ is implied by $C_2^+(\vec{x}, \vec{y})$; else we call the equivalence class *free*. Let us consider first the variables in the bound equivalence classes. Each such variable x_i can be expressed as $x_i = c$ (or $x_i = \mathsf{n} - c$), and/or as $x_i = y_{j_1} + c_1, x_i = y_{j_2} + c_2, \ldots$ in $C_2^+(\vec{x}, \vec{y})$; the conditions $x_i = c$ and $x_i = \mathsf{n} - c$ cannot occur simultaneously, since $C(\vec{x}, \vec{y})$ is satisfiable. Choose arbitrarily one such way of expressing x_i, i.e. define $A_i(\vec{z}, \vec{y})$ to be either c (or $\mathsf{n} - c$), or $y_{j_1} + c_1$, or $y_{j_1} + c_2$, ... Equate all expressions for x_i, and let $D_i(\vec{y})$ be the conjunction of all such equations; i.e. $D_i(\vec{y})$ will be $c = y_{j_1} + c_1 \wedge c = y_{j_2} + c_2 \wedge c = y_{j_3} + c_3 \wedge \ldots$, when a condition of the form $x_i = c$ can be inferred from $C_2^+(\vec{x}, \vec{y})$, or simply $y_{j_1} + c_1 = y_{j_2} + c_2 \wedge y_{j_1} + c_1 = y_{j_3} + c_3 \wedge \ldots$, when no condition of the form $x_i = c$, or of the form $x_i = \mathsf{n} - c$, can be inferred from $C_2^+(\vec{x}, \vec{y})$. Next, let k be the number of free equivalence classes; we proceed here as in the proof of Proposition 6.6, namely we number these equivalence classes $1, 2, \ldots, k$, choose a representative from each class, and rename these representatives $z_1, \ldots, z_k$. Then, each variable x_i in a free equivalence class can be uniquely expressed as a function of the representative z_{α_i} of its equivalence class, α_i, namely $x_i = z_{\alpha_i} + c_i$. Define $A_i(\vec{z}, \vec{y}) \stackrel{\text{def}}{=} z_{\alpha_i} + c_i$. Finally define $V(\vec{y})$ to be the variable affine abstract expression:

$$V(\vec{y}) \stackrel{\text{def}}{=} \{\langle A_1(\vec{z}, \vec{y}), \ldots, A_p(\vec{z}, \vec{y})\rangle \mid \bigwedge_{i=1,p} (0 \leq A_i(\vec{z}, \vec{y}) \leq \mathsf{n}) \wedge C_2^-(A_1(\vec{z}, \vec{y}), \ldots, A_p(\vec{z}, \vec{y}))\}$$

Let U' be the maximal affine abstract expression associated to $V(\vec{y})$, by Proposition 7.4. Then define:

$$U \stackrel{\text{def}}{=} \{\vec{y} \mid C_1(\vec{y}) \wedge \bigwedge_i D_i(\vec{y})\} \cap U'$$

U is an affine a.e. by Corollary 6.9.

Conversely, let $U, V(\vec{y})$ be associated affine abstract expressions. Applying Proposition 6.6 directly, we obtain a dual-affine abstract expression $\{\vec{y} \mid \vec{y} = 0, \mathsf{n}; C(\vec{y})\}$ equivalent to U. Next we consider the variable affine abstract expression $V(\vec{y}) = \{\vec{A}(\vec{z}, \vec{y}) \mid \vec{z} = 0, \mathsf{n}; C^-(\vec{z}, \vec{y})\}$. As in Proposition 6.6, we introduce new variables $x_1, \ldots, x_p$, and construct the equalities $x_1 = A_1, \ldots, x_p = A_p$. First consider all those equalities $x_i = A_i$ in which A_i mentions some parameter variable, say z_j; rearrange each of them in the form $z_j = x_i + c_i$. In this way we express every parameter z_j as a function of one, or several x_i's; if there are more than one, we equate these expressions pairwise, and call $C_1(\vec{x})$ the conjunction of these conditions. In addition, we impose the conditions $x_i + c_i \geq 0$ and $x_i + c_i \leq \mathsf{n}$ (see Definition 4.3); call $C_2(\vec{x})$ the resulting conditions. Next we consider all conditions $x_i = A_i$ in which A_i is a constant c, or $\mathsf{n} - c$, or $y_j + c$; call $C_3(\vec{x}, \vec{y})$ the conjunction of these conditions. Finally we define $C_4(\vec{x}, \vec{y})$ to be the condition $C^-(\vec{z}, \vec{y})$ in which every variable z_j is replaced with its corresponding $x_i + c_i$ (choose arbitrarily one such simple abstract expression, if there are more for the same z_j). Finally the equivalent variable dual-affine abstract expression is:

$$A \stackrel{\text{def}}{=} \{\langle x_1, \ldots, x_k\rangle \mid C \wedge C_1 \wedge C_2 \wedge C_3 \wedge C_4\}$$

□

Let $V(\vec{y}) = \{\langle A_1, \ldots, A_p\rangle \mid \vec{z} = 0, \mathsf{n}; C^-(\vec{z}, \vec{y})\}$ be a variable affine abstract expression with k dimensions. For $i = 1, \ldots, p$, if $A_i(\vec{z}, \vec{y})$ depends on some parameter z_j, we say that $V(\vec{y})$ is **free along the dimension** i. Else (if $A_i(\vec{z}, \vec{y})$ is constant, or $\mathsf{n} - c$, or $y_j + c$), we say that $V(\vec{y})$ is **bound** at the dimension i. In Example 7.2, V is bound along the dimensions 1 and 3, and free along dimension 2.

Proposition 7.5 essentially identifies any satisfiable conjunctive condition $C(\vec{x}, \vec{y})$ with an associated pair $U, V(\vec{y})$; the condition $C(\vec{x}, \vec{y})$ is equivalent to the condition $\vec{y} \in U \wedge \vec{x} \in V(\vec{y})$. For some other condition $C'(\vec{x}, \vec{y})$ we will say that we "intersect" $U, V(\vec{y})$ with $C'(\vec{x}, \vec{y})$, meaning the associated pair $U', V'(\vec{y})$ corresponding to the condition $C(\vec{x}, \vec{y}) \wedge C'(\vec{x}, \vec{y})$, in the sense of Proposition 7.4. When $C'(\vec{x}, \vec{y})$ is independent of $\vec{x}$, then $V'(\vec{y}) = V(\vec{y})$. When $C'(\vec{x}, \vec{y})$ is independent of $\vec{y}$, we will have in general $U' \neq U$ and $V'(\vec{y}) \neq V(\vec{y})$, because some positive condition $x_i = x_j + c$ in $C'(\vec{x}, \vec{y})$ may impose some conditions on the variables $\vec{y}$, when x_i and/or x_j are bound to some variable y in $V(\vec{y})$. However, when $C'(\vec{x}, \vec{y}) = (x_i = x_j + c)$, and $V(\vec{y})$ is free along both dimensions i and j, then $U' = U$, because the condition $x_i = x_j + c$ does not affect the variables $\vec{y}$.

7.2 Cliques for Conjunctions of Simple Conditions

Let $C(\vec{x}, \vec{x}', \vec{y})$ be a condition, $\vec{x} = \langle x_1, \ldots, x_p \rangle, \vec{x}' = \langle x'_1, \ldots, x'_p \rangle, \vec{y} = \langle y_1, \ldots, y_q \rangle$.

Definition 7.6 *Let $n \geq 0$. For some $\vec{y} \in [n]^q$, we say that a sequence $s = [\vec{x}^1, \ldots, \vec{x}^m]$ of distinct elements from $[n]^p$ is* **a** m**-clique for** C **at** $\vec{y}$, *if for all i, j, $1 \leq i < j \leq m$, $C(\vec{x}^i, \vec{x}^j, \vec{y})$ is true. We call m the* **size** *of the clique s.*

We will prove that for any condition $C(\vec{x}, \vec{x}', \vec{y})$ the following **Clique Property** holds:

The Clique Property for condition $C(\vec{x}, \vec{x}', \vec{y})$**:**

1. for all $n \geq 0$ and $\vec{y} \in [n]^q$, all cliques for C at $\vec{y}$ have size $O(1)$, or
2. there is some affine abstract expression U such that $\forall n \geq 0, \forall \vec{y} \in [\![U]\!]^{[n]}$, there are cliques of size at least $\Omega(n)$ for C at $\vec{y}$.

In this Section we will prove this property for the simpler case when C is a conjunction of simple conditions. Surprisingly, in this case item 2 of the Clique Property will hold whenever C has cliques of at least size 4 for arbitrarily large n.

Example 7.7 *Let $C(\langle x_1, x_2, x_3 \rangle, \langle x'_1, x'_2, x'_3 \rangle) = (x_2 = x'_1 + 1)$. Here $p = 3$ and $q = 0$. Condition 2 of the Clique Property holds, since the sequence $[\langle 0, 1, 0 \rangle, \langle 0, 1, 1 \rangle, \langle 0, 1, 2 \rangle, \ldots, \langle 0, 1, n \rangle]$ is a clique of size $n+1$; indeed $C(\langle 0, 1, i \rangle, \langle 0, 1, j \rangle)$ is true, $\forall i, j$, $1 \leq i < j \leq n$.*

Example 7.8 *Let $C(\langle x_1, x_2, x_3 \rangle, \langle x'_1, x'_2, x'_3 \rangle) = (x_2 = x'_1 + 1 \wedge x_3 = x'_2 + 1)$. Here $p = 3, q = 0$, and condition 1 of the Clique Property holds since any clique has at most size 3. E.g. $\langle 1, 1, 2 \rangle, \langle 0, 1, 2 \rangle, \langle 0, 1, 1 \rangle$ is such a clique. Indeed, all cliques of size 3 for C have the form $\langle \beta, \alpha, \alpha + 1 \rangle, \langle \alpha - 1, \alpha, \alpha + 1 \rangle, \langle \alpha - 1, \alpha, \gamma \rangle$, and the reader may convince himself that longer cliques do not exist, due to the constraint that all "middle" elements (those which are not the first or the last one) have the form $\langle \alpha - 1, \alpha, \alpha + 1 \rangle$. Recall that cliques must have* distinct *elements.*

Example 7.9 *Let $C(\langle x_1, x_2 \rangle, \langle x'_1, x'_2 \rangle) = (x_1 = 5 \wedge x_1 = n - 5)$. For $n = 10$ we can find "long" cliques for C, e.g. $\langle 5, 0 \rangle, \langle 5, 1 \rangle, \ldots, \langle 5, 10 \rangle$, but for $n \neq 10$ there are no cliques at all. Condition 1 holds in this example.*

For the rest of this Subsection let $C(\vec{x}, \vec{x}', \vec{y})$ be a satisfiable conjunction of simple conditions. Since each simple condition can mention at most two variables, $C(\vec{x}, \vec{x}', \vec{y})$ can be expressed as: $C(\vec{x}, \vec{x}', \vec{y}) = D(\vec{x}, \vec{y}) \wedge D'(\vec{x}', \vec{y}) \wedge E(\vec{x}, \vec{x}')$, where $E(\vec{x}, \vec{x}')$ contains exactly those conditions which mention *both* one variable from $\vec{x}$ and one from $\vec{x}'$. The conditions mentioning only variables in $\vec{y}$ may be included arbitrarily in D or D'.

The next Lemma is the first step in the proof of the Clique Property. It roughly says that for a given $C(\vec{x}, \vec{x}', \vec{y})$, we can find an associated pair of affine abstract expressions $U, V(\vec{y})$ such that $[\vec{x}^1, \ldots, \vec{x}^m]$ is a clique for $C(\vec{x}, \vec{x}', \vec{y})$ at $\vec{y}$ iff it is a clique for $E(\vec{x}, \vec{x}')$, $\vec{y} \in U$, and $\vec{x}^2, \ldots, \vec{x}^{m-1} \in V(\vec{y})$. Unfortunately the two statements are not entirely equivalent; we are forced to consider more relaxed versions of the two implications, making the statement of the next Lemma rather cumbersome. First we need a definition:

Definition 7.10 *We say that C has m-cliques* **for arbitrarily large** n, *iff* $\forall n_0, \exists n \geq n_0$, $\exists \vec{y} \in [n]^q$, $\exists \vec{x}^1, \ldots, \vec{x}^m \in [n]^p$, *s.t.* $[\vec{x}^1, \ldots, \vec{x}^m]$ *is an m-clique for C at $\vec{y}$.*

Lemma 7.11 *Suppose that $C(\vec{x}, \vec{x}', \vec{y})$ has cliques of size 3, for arbitrarily large n. Then there exists associated affine abstract expressions U and $V(\vec{y})$, such that for every $n \geq 0$, every $\vec{y}$, and every sequence $s = [\vec{x}^1, \ldots, \vec{x}^m]$, the following hold:*

1. *If s is a clique for $C(\vec{x}, \vec{x}', \vec{y})$ at $\vec{y}$, and $m \geq 3$, then the following hold:*
   (a) it is also a clique for $E(\vec{x}, \vec{x}')$
   (b) $y \in U$
   (c) $\vec{x}^2, \ldots, \vec{x}^{m-1} \in V(\vec{y})$.
2. *If s is a clique for $E(\vec{x}, \vec{x}')$, $\vec{y} \in U$, and $\vec{x}^1, \ldots, \vec{x}^m \in V(\vec{y})$, then s is also a clique for $C(\vec{x}, \vec{x}', \vec{y})$ at $\vec{y}$.*

Proof. Consider the condition $F(\vec{x}, \vec{y}) = D(\vec{x}, \vec{y}) \wedge D'(\vec{x}, \vec{y})$ (i.e. we substitute $\vec{x}'$ with $\vec{x}$ in $D'(\vec{x}', \vec{y})$). For an arbitrary large n, let $[\vec{x}^1, \vec{x}^2, \vec{x}^3]$ be a clique for $C(\vec{x}, \vec{x}', \vec{y})$ at some $\vec{y}$. Since $C(\vec{x}^1, \vec{x}^2, \vec{y}) = D(\vec{x}^1, \vec{y}) \wedge D'(\vec{x}^2, \vec{y}) \wedge E(\vec{x}^1, \vec{x}^2)$ is true, it follows that $D'(\vec{x}^2, \vec{y})$ is true. Similarly $D(\vec{x}^2, \vec{y})$ is true, because of $C(\vec{x}^2, \vec{x}^3, \vec{y})$. So $F(\vec{x}^2, \vec{y})$ is true. Hence F is satisfiable, in the sense of Section 6, because n may be chosen arbitrarily large. By Proposition 7.5, there exists associated affine a.e. U and $V(\vec{y})$, such that $F(\vec{x}, \vec{y})$ is equivalent to $\vec{y} \in U$ and $\vec{x} \in V(\vec{y})$. We prove that U and $V(\vec{y})$ satisfy items 1 and 2 above.

Item 1a is obviously true. To prove 1b and 1c, suppose s is a clique for $C(\vec{x}, \vec{x}', \vec{y})$ at $\vec{y}$. Consider i s.t. $2 \leq i \leq m-1$. Since $C(\vec{x}^{i-1}, \vec{x}^i, \vec{y})$ is true, it follows that $D'(\vec{x}^i, \vec{y})$ is true. Similarly, from $C(\vec{x}^i, \vec{x}^{i+1}, \vec{y})$, we conclude that $D(\vec{x}^i, \vec{y})$ is true. Hence $F(\vec{x}^i, \vec{y})$ is true, so $\vec{y} \in U$, and $\vec{x}^i \in V(\vec{y})$.

To prove item 2, suppose s is a clique for $E(\vec{x}, \vec{x}')$ and $\vec{y} \in U$. We have to prove that $C(\vec{x}^i, \vec{x}^j, \vec{y})$, i.e. $D(\vec{x}^i, \vec{y}) \wedge D'(\vec{x}^j, \vec{y}) \wedge E(\vec{x}^i, \vec{x}^j)$, is true for $1 \leq i < j \leq n$. Obviously $E(\vec{x}^i, \vec{x}^j)$ is true. From $\vec{y} \in U$ and $\vec{x}^i \in V(\vec{y})$, we conclude that $F(\vec{x}^i, \vec{y})$ is true, hence $D(\vec{x}^i, \vec{y})$ is true. Similarly for $D'(\vec{x}^j, \vec{y})$. $\square$

Definition 7.12 *Let $\vec{x}, \vec{x}' \in [n]^p$. We define their* **distance** $dist(\vec{x}, \vec{x}') = \min(\{|x_k - x'_l| \mid 1 \leq k, l \leq p\})$.

Note that *dist* is not a true topological distance, e.g. we may have $dist(\vec{x}, \vec{x}') = 0$ although $\vec{x} \neq \vec{x}'$.

Finally we can prove the Clique Property, for conditions which are conjunctions of simple conditions.

Proposition 7.13 (The Clique Property for conjunctive conditions) *Let $C(\vec{x}, \vec{x}', \vec{y})$ be a conjunction of simple conditions. Then one of the following holds:*

1. *There exists some $m > 0$ s.t. $\forall n$, $\forall \vec{y} \in [n]^q$, any clique for $C(\vec{x}, \vec{x}', \vec{y})$ at $\vec{y}$ has size $\leq m$. In other words, the cliques for $C(\vec{x}, \vec{x}', \vec{y})$ have size $O(1)$.*

2. *There exists an affine a.e. U and two integer constants $\gamma, \delta > 0$ such that $\forall n \geq 0$, $\forall \vec{y} \in U$, there exists a clique of length $\lfloor \frac{n-\delta}{\gamma} \rfloor$ for $C(\vec{x}, \vec{x}', \vec{y})$ at $\vec{y}$. In other words, $\forall \vec{y} \in U$ there are cliques for $C(\vec{x}, \vec{x}', \vec{y})$ at $\vec{y}$, of size $\Omega(n)$.*

Proof. Suppose Condition 1 does not hold. Then C has cliques of size 4 for arbitrarily large n. Indeed, if not, then there is some n_0 such that for every $n \geq n_0$ all cliques have size ≤ 3. But then it suffices to pick $m = \max(3, n_0^p)$, and condition 1 is satisfied.

Then by Lemma 7.11, we construct U and $V(\vec{y})$ such that, by item 2, for any $\vec{y} \in U$, any clique for $E(\vec{x}, \vec{x}')$, which is included in $V(\vec{y})$ is also a clique for C at $\vec{y}$. By item 1 of the same Lemma, whenever we have a clique of length 4, $\vec{x}^1, \vec{x}^2, \vec{x}^3, \vec{x}^4$, for C at $\vec{y}$, we also have $\vec{y} \in U$ and $\vec{x}^2, \vec{x}^3 \in V(\vec{y})$. So $V(\vec{y})$ has at least 1 dimension, because C has cliques of size 4, for arbitrarily large n.

Now we consider the condition $E(\vec{x}, \vec{x}')$, which we split into *positive* and *negative* simple conditions, $E(\vec{x}, \vec{x}') = E^+(\vec{x}, \vec{x}') \wedge E^-(\vec{x}, \vec{x}')$. First we prove that for any clique $[\vec{x}^1, \vec{x}^2, \vec{x}^3]$ for E^+, $E^+(\vec{x}^2, \vec{x}^2)$ is true. Indeed, consider some condition $x_k = x_l' + c$ in E^+. Since $E^+(\vec{x}^1, \vec{x}^2)$, $E^+(\vec{x}^1, \vec{x}^3)$ and $E^+(\vec{x}^2, \vec{x}^3)$ are true, we have $x_k^1 = x_l^2 + c$, $x_k^1 = x_l^3 + c$, and $x_k^2 = x_l^3 + c$. So we deduce $x_k^2 = x_l^2 + c$ and hence $E^+(\vec{x}^2, \vec{x}^2)$. In particular this proves that $E^+(\vec{x}, \vec{x})$ is satisfiable.

Next we will construct $U' \subseteq U$ and $V'(\vec{y}) \subseteq V(\vec{y})$ such that $\forall y \in U'$, any sequence included in $V'(\vec{y})$ is a clique for E^+. For this, we start by taking $U' = U$ and $V'(\vec{y}) = V(\vec{y})$, and shrink U' and $V'(\vec{y})$ while processing every simple condition $x_k = x_l' + c$ in E^+. For each such condition, exactly one of the four cases listed below will hold. We will process the conditions in two phases. In the first phase we process *one by one* those conditions for which one of the cases 1,2,3 holds. During this phase the following two invariants are preserved:

INV1 For every $n \geq 0$ and every m-clique, $m \geq 3$, $[\vec{x}^1, \ldots, \vec{x}^m]$ for $C(\vec{x}, \vec{x}', \vec{y})$ at $\vec{y}$, we have: $\vec{y} \in U'$ and $\vec{x}^2, \ldots, \vec{x}^{m-1} \in V'(\vec{y})$. Since $C(\vec{x}, \vec{x}', \vec{y})$ has 4-cliques for arbitrarily large n, this will ensure that U' is non-empty and that $V'(\vec{y})$ has ≥ 1 dimensions.

INV2 Let $E'(\vec{x}, \vec{x}')$ be the conjunction of the remaining conditions in $E(\vec{x}, \vec{x}')$, not yet processed. Then $\forall n \geq 0$, $\forall \vec{y} \in U'$, any clique $[\vec{x}^1, \ldots, \vec{x}^m]$ for $E'(\vec{x}, \vec{x}')$, for which $\vec{x}^1, \ldots, \vec{x}^m \in V'(\vec{y})$, is also a clique of $C(\vec{x}, \vec{x}', \vec{y})$ at $\vec{y}$.

Initially, when $U' = U$ and $V'(\vec{y}) = V(\vec{y})$, both conditions are satisfied by Lemma 7.11.

In the second phase we process *in one step* all those conditions for which Case 4 below holds. Here we will only shrink $V'(\vec{y})$. The invariant INV2 above will still hold (with $E'(\vec{x}, \vec{x}') = E^-(\vec{x}, \vec{x}')$), but invariant INV1 will not. We will use another argument to show that $V'(\vec{y})$ still has ≥ 1 dimensions.

So let $x_k = x_l' + c$ be one of the simple conditions in E^+. One of the following four cases may occur.

1. $V'(\vec{y})$ is *bound* along both dimensions k and l (see Subsection 7.1), i.e. the expressions A_k and A_l in $V'(\vec{y})$ are either constants or mention some variable from $\vec{y}$ (but not a parameter z of the affine a.e. $V'(\vec{y})$). Then we impose the condition $A_k = A_l + c$, which may shrink U' (because it may impose some constraints on the $\vec{y}$'s). To check INV1, let $[\vec{x}^1, \ldots, \vec{x}^m]$ be a clique of length $m \geq 3$ for C at $\vec{y}$. By induction hypothesis $\vec{y} \in U'$, and, since $\vec{x}^2 \in V'(\vec{y})$ and $E^+(\vec{x}^2, \vec{x}^2)$ is true, it follows that $\vec{y}$ also satisfies $A_k = A_l + c$ (a condition only on $\vec{y}$). To check INV2, pick some $\vec{y} \in U'$ satisfying $A_k = A_l + c$, and let $[\vec{x}^1, \ldots, \vec{x}^m]$ be a clique for the remaining unprocessed condition $E'(\vec{x}, \vec{x}')$, not including the current condition $x_k = x'_l + c$. But $\vec{x}^1, \ldots, \vec{x}^m$ are in $V(\vec{y})$, which is *bound* along the dimensions k and l. Hence all m vectors $\vec{x}^1, \ldots, \vec{x}^m$ will have the same value d on position k, and the same value d' on position l. Moreover, since $\vec{y}$ satisfies $A_k = A_l + c$, we have $d = d' + c$, so $[\vec{x}^1, \ldots, \vec{x}^m]$ will be a clique for $(x_k = x'_l + c) \wedge E'(\vec{x}, \vec{x}')$. Finally we may apply induction hypothesis to argue that it is also a clique for $C(\vec{x}, \vec{x}', \vec{y})$ at $\vec{y}$.

2. $V'(\vec{y})$ is bound along dimension k and free along dimension l, or vice versa. As in the previous case, here we also impose the condition $A_k = A_l + c$, but which now will shrink the affine a.e. $V'(\vec{y})$. We invite the reader to convince himself that INV1 and INV2 are preserved. After this, V' will be bound along both dimensions k and l.

3. $V'(\vec{y})$ is free along both dimensions k and l, and has the same parameter variable z at these two dimensions. In this case $A_k = A_l + c$ is a tautology and U', $V'(\vec{y})$ will remain unchanged.

4. $V'(\vec{y})$ is free along both dimensions k and l, and has different parameter variables at these dimensions. As explained earlier, we treat all remaining such conditions $x_{k_1} = x'_{l_1} + c_1, \ldots, x_{k_\kappa} = x'_{l_\kappa} + c_\kappa$ at the end, *after* treating the conditions which fall into one of the other three cases. Here we need a particular clique of length 4, say $\vec{x}^1_0, \vec{x}^2_0, \vec{x}^3_0, \vec{x}^4_0$, for some particular $n_0 \geq 0$. One observes easily that, for any $\iota = 1, \ldots, \kappa$, $\vec{x}^1_0, \vec{x}^2_0, \vec{x}^3_0$ all have the same number, say d_ι on position k_ι. Then we intersect U' and $V'(\vec{y})$ with $V'' \stackrel{\text{def}}{=} \{\vec{x} \mid \bigwedge_\iota (x_{k_\iota} = d_\iota \wedge x_{l_\iota} = d_\iota - c_\iota)\}$. Note that this may affect U indirectly: even if the dimensions k_ι and l_ι of $V'(\vec{y})$ are free, they may still be subject to inequality constraints involving some $\vec{y}$. This may impose some additional negative conditions on U', but which do not reduce its number of dimensions. Also the new $V'(\vec{y})$ will still have at least ≥ 1 dimensions, because for that particular n_0 and that particular $\vec{y}_0$ in contains both $\vec{x}_2$ and $\vec{x}_3$. We prove now that INV2 holds after this step (here $E'(\vec{x}, \vec{x}')$ is $E^-(\vec{x}, \vec{x}')$). Indeed, for $\vec{y} \in U'$ and some arbitrary clique $[\vec{x}^1, \ldots, \vec{x}^m]$ for $E^-(\vec{x}, \vec{x}')$, if $\vec{x}^1, \ldots, \vec{x}^m$ are in $V'(\vec{y}) \cap V''$, then all of them have the value d_ι on position k_ι, and the value $d_\iota - c_\iota$, on position l_ι, for $\iota = 1, \ldots, \kappa$. Hence the condition $\bigwedge_\iota (x^i_{k_\iota} = x^j_{l_\iota} + c_\iota)$ holds trivially, and $[\vec{x}^1, \ldots, \vec{x}^m]$ will be also a clique for $E^-(\vec{x}, \vec{x}') \wedge \bigwedge_\iota (x_{k_\iota} = x'_{l_\iota} + c_\iota)$. Then, by induction hypothesis, it will also be a clique for $C(\vec{x}, \vec{x}', \vec{y})$.

Finally process, one by one, each of the negative conditions $x_i \neq x'_j + c$ in E^-, and shrink $U', V'(\vec{y})$ while maintaining INV2. We distinguish the same four cases as above. Cases 1 and 2 are treated similarly, but without decreasing the number of dimensions of U' or $V'(\vec{y})$. We do not process those conditions falling into the cases 3 and 4. Here the condition $x_i \neq x'_j + c$ translates to $z_{i'} \neq z_{j'} + c'$. The parameters $z_{i'}$ and $z_{j'}$ will be the same in case 3 and different in case 4. We define the number γ to be:

$$\gamma \stackrel{\text{def}}{=} 1 + \max(\{|c'| \mid z_{i'} \neq z_{j'} + c' \text{ is a condition in } E^- \text{ as described above}\})$$

So in the end, we have some affine a.e. U', variable affine a.e. $V'(\vec{y}) = \{\vec{A}(\vec{z}, \vec{y}) \mid \vec{z} = 0, \mathsf{n}; C'(\vec{z}, \vec{y})\}$, with $k > 0$ dimensions, and remaining negative conditions $E'(\vec{x}, \vec{x}')$, each condition involving only free dimensions of $V'(\vec{y})$. Moreover, the invariant INV2 holds.

Consider some $n \geq 0, \vec{y} \in U'$, and a sequence $s = [\vec{x}^1, \dots, \vec{x}^m]$ included in $V'(\vec{y})$. Hence $\vec{x}^1 = \vec{A}(\vec{z}^1), \dots, \vec{x}^m = \vec{A}(\vec{z}^m)$, for some sequence $\vec{z}^1, \dots, \vec{z}^m \in [n]^k$. If $dist(\vec{z}^i, \vec{z}^j) \geq \gamma$, $\forall i, j, 1 \leq i < j \leq m$, then s is a clique for $E'(\vec{x}, \vec{x}')$ and therefore (by INV2) it is a clique for $C(\vec{x}, \vec{x}', \vec{y})$. So it only remains to argue that we can find "long" sequences $[\vec{z}^1, \dots, \vec{z}^m]$ for which $dist(\vec{z}^i, \vec{z}^j) \geq \gamma$. Intuitively we will choose $\vec{z}^1, \dots, \vec{z}^m$ along a diagonal in the space $[n]^k$, at distance γ from each other; we should be able to find $\Omega(\frac{n}{\gamma})$ such z's. More precisely, observe that the only conditions imposed by $C'(\vec{z}, \vec{y})$ (the negative condition of $V'(\vec{y})$) on $\vec{z}$ are negative. Hence, for n large enough, it can be satisfied by a "small" $\vec{z}^1$, i.e. there is some constant δ, independent of n and $\vec{y}$, such that $\forall n \geq 0, \forall y \in U'$, $\exists \vec{z}^1 \in [n]^k$ s.t. $C(\vec{z}, \vec{y})$ is true and $z_1^1 \leq \delta, z_2^1 \leq \delta, \dots z_k^1 \leq \delta$. Then define $\vec{z}^i \stackrel{\text{def}}{=} \vec{z}^1 + \langle (i-1)\gamma, \dots, (i-1)\gamma \rangle$, $i = 2, \lfloor \frac{n-\delta}{\gamma} \rfloor$. Not all of them will satisfy $C'(\vec{z}, \vec{y})$. However, let l be the number of conditions in $C'(\vec{z}, \vec{y})$ (recall, all are negative conditions). We show that, after eliminating at most l elements from the sequence $\vec{z}^1, \vec{z}^2, \vec{z}^3, \dots$, we get a subsequence $\vec{z}^{i_1}, \dots, \vec{z}^{i_m}$, with $m = \lfloor \frac{n-\delta}{\gamma} \rfloor - l = \lfloor \frac{n-(\delta+l\gamma)}{\gamma} \rfloor = \Omega(n)$, satisfying $C'(\vec{z}, \vec{y})$. Indeed, all conditions of the form $z_i \neq z_j + c$ are satisfied by $\vec{z}^1$, and hence they are satisfied by $\vec{z}^2, \vec{z}^3, \dots$, while each of remaining conditions, of the form $z_i \neq y_j + c$, invalidates at most one of $\vec{z}^2, \vec{z}^3, \dots$. So we obtain a clique $\vec{x}^1 \stackrel{\text{def}}{=} \vec{A}(\vec{z}^{i_1}), \dots, \vec{x}^m \stackrel{\text{def}}{=} \vec{A}(\vec{z}^{i_m})$ for $C(\vec{x}, \vec{x}', \vec{y})$ at $\vec{y}$, of length $\Omega(n)$. □

7.3 Cliques for Arbitrary Conditions

To generalize Proposition 7.13 to arbitrary conditions D, we recall a known fact in graph theory:

Theorem 7.14 *([Bol79], pp. 104, Theorem 1) Let G be a complete, undirected graph with $C_{2m-2}^{m-1} (= \frac{(2m-2)!}{(m-1)!})$ vertices. Suppose we color each edge of G either with red or with blue. Then there exists a complete subgraph of G, with m vertices, having all edges colored with the same color.*

This gives us immediately:

Lemma 7.15 *Let $C(\vec{x}, \vec{x}', \vec{y}) = C_1(\vec{x}, \vec{x}', \vec{y}) \vee C_2(\vec{x}, \vec{x}', \vec{y})$ be some condition with $\vec{x} = \langle x_1, \dots, x_p \rangle$, $\vec{x}' = \langle x'_1, \dots, x'_p \rangle$, $\vec{y} = \langle y_1, \dots, y_q \rangle$. If C admits arbitrarily long cliques, then either $C_1(\vec{x}, \vec{x}', \vec{y})$, or $C_2(\vec{x}, \vec{x}', \vec{y})$ admit arbitrarily long cliques.*

Proof. Let $m \geq 0$. We show that either $C_1(\vec{x}, \vec{x}', \vec{y})$, or $C_2(\vec{x}, \vec{x}', \vec{y})$, has a clique of length m. Choose $m' = C_{2m-2}^{m-1}$. By hypothesis there exists $n \geq 0$ and $\vec{y} \in [n]^q$ for which a clique $[\vec{x}^1, \dots, \vec{x}^{m'}]$ for $C(\vec{x}, \vec{x}', \vec{y})$ at $\vec{y}$ exists. Consider the complete undirected graph having $\{1, 2, \dots, m'\}$ as the set of nodes. For every pair of nodes i, j, $i < j$, color the edge (i, j) red, if $C_1(\vec{x}^i, \vec{x}^j, \vec{y})$ is true, and color it blue, if $C_2(\vec{x}^i, \vec{x}^j, \vec{y})$ is true (color it red, when both conditions are true). By Theorem 7.14 there exists a complete subgraph with m vertices, say $\{\alpha_1, \dots, \alpha_m\} \subseteq \{1, 2, \dots, m'\}$, s.t. all edges (α_i, α_j), $i < j$, have the same color, say blue. Then $[\vec{x}^{\alpha_1}, \dots, \vec{x}^{\alpha_m}]$ is a clique for $C_2(\vec{x}, \vec{x}', \vec{y})$. □

So we can generalize Proposition 7.13 to:

Proposition 7.16 *Let $C(\vec{x}, \vec{x}', \vec{y})$ be some condition with $\vec{x}, \vec{x}'$ of the same length. Then either all cliques for C have length $O(1)$, or there is some affine a.e. U such that $\forall \vec{y} \in U$, there exists a clique of length $\Omega(n)$ for C at $\vec{y}$.*

Proof. We write C in disjunctive normal form $C = C_1 \vee \ldots \vee C_k$, where $C_1, \ldots, C_k$ are conjunctive conditions, and assume that C admits arbitrarily long cliques. By repeatedly applying Lemma 7.15, we conclude that for some i, the condition C_i admits arbitrarily long cliques. Here we apply Proposition 7.13. $\square$

7.4 Applying powerset to Abstract Expressions

Theorem 7.17 (Evaluation Theorem with powerset) *Let $A(\vec{y})$ be some abstract expression of type t_1, $C(\vec{y})$ some condition and $f : t_1 \to t_2$ some expression in $\mathcal{NRA}(\mathsf{powerset})$. Then one of the following holds:*

1. *There is some abstract expression $A'(\vec{y})$ such that:*

$$\forall n.\forall \rho \in \mathsf{Env}^{[n]}.[\![C]\!]^{[n]}(\rho) = \mathsf{T} \Rightarrow f([\![A]\!]^{[n]}(\rho)) \Downarrow [\![A']\!]^{[n]}(\rho)$$

   We abbreviate this with $C \Rightarrow (f(A) \Downarrow A')$. More, in this case $\exists m_f \geq 0$ such that $\forall m \geq m_f$, $C \Rightarrow (f_m(A) \Downarrow A')$, where f_m is the m'th approximation of f described in Section 3 (obtained from f by replacing each occurrence of powerset *with the approximation* $\mathsf{powerset}_m$*), and, hence, does not mention* powerset.

2. *The complexity of $C \Rightarrow (f(A) \Downarrow)$ is $\Omega(2^{cn})$, for some $c > 0$, i.e. $\exists c > 0$ such that for arbitrarily large n, $\exists \rho \in \mathsf{Env}^{[n]}$ s.t. $[\![C]\!]^{[n]}(\rho) = \mathsf{T}$ and $\mathsf{complex}(f, [\![A]\!]^{[n]}(\rho)) > 2^{cn}$.*

Intuitively, Theorem 7.17 generalizes Theorem 5.2 to expressions involving powerset. The intuition is that the computation of $\mathsf{powerset}(A) \Downarrow$ may have exponential or polynomial complexity, depending on the values of the free variables in A. The condition $C(\vec{y})$ captures exactly these informations. E.g. when $A = \{\langle x, y\rangle \mid x = 0, \mathsf{n}; y \neq 1\}$, then for $C(y) = (y = 1)$ we have $C \Rightarrow (\mathsf{powerset}(A) \Downarrow \{\emptyset\})$, while for $C(y) = (y \neq 1)$ the complexity of $C \Rightarrow (\mathsf{powerset}(A) \Downarrow)$ is exponential.

Proof. The proof is done by induction on the structure of f. All cases, except powerset, are extensions of those in Theorem 5.2. We discuss these cases first, then focus on the powerset case.

When f is one of $\mathsf{id}, !, \pi_1, \pi_2, \eta, \mu, \Delta_{\mathsf{left}}, \emptyset, \cup, =, \mathsf{empty}, \mathsf{true}, \mathsf{false}$, the claim follows directly from Theorem 5.2. Consider now the composition case, $g \circ f$. To "compute" $C \Rightarrow (g \circ f(A) \Downarrow)$, we start by applying induction hypothesis to $C \Rightarrow (f(A) \Downarrow)$. If case 2 holds here, i.e. the complexity is exponential, then the complexity of $C \Rightarrow (g \circ f(A) \Downarrow)$ is exponential too, and we are done. So assume case 1 holds, i.e. there exists some abstract expression $A'(\vec{y})$ such that $C \Rightarrow (f(A) \Downarrow A')$. Now we apply induction hypothesis to $C \Rightarrow (g(A') \Downarrow)$. If the complexity of this evaluation is exponential, then the complexity of $C \Rightarrow (g \circ f(A) \Downarrow)$ is exponential too, and we are done. So assume case 1 holds for $C \Rightarrow (g(A') \Downarrow)$, hence there exists $A''(\vec{y})$ such that $C \Rightarrow (g(A') \Downarrow A'')$. Obviously we have $C \Rightarrow (g \circ f(A) \Downarrow A'')$. Moreover, in this last case, we get numbers m_f and m_g such that, under condition C, $\forall m \geq m_f$ f_m "approximates" f, and $\forall m \geq m_g$ g_m "approximates" g. It suffices to take $m_{g \circ f} \stackrel{\text{def}}{=} \max(m_f, m_g)$, and we get that $(g \circ f)_m = g_m \circ f_m$ "approximates" $g \circ f$ under condition C for all $m \geq m_{g \circ f}$.

The case $\langle f, g\rangle$ is handled similar to the case $g \circ f$.

Consider now the case $\mathsf{map}(f)$. To compute $C \Rightarrow (\mathsf{map}(f)(A) \Downarrow)$, we consider several cases for A as in Theorem 5.2. The most interesting one is when $A = \{A' \mid \vec{x} = 0, \mathsf{n}; C'(\vec{x}, \vec{y})\}$, which we illustrate here. Namely we first apply induction hypothesis to $C \wedge C' \Rightarrow (f(A') \Downarrow)$. If its complexity is exponential, then so is the complexity of $C \Rightarrow (\mathsf{map}(f)(A) \Downarrow)$. Else, we find some A'' s.t. $C \wedge C' \Rightarrow (f(A') \Downarrow A'')$. Then we conclude that $C \Rightarrow (\mathsf{map}(f)(\{A' \mid \vec{x} = 0, \mathsf{n}; C'\}) \Downarrow \{A'' \mid \vec{x} = 0, \mathsf{n}; C'\})$. Moreover, whenever f_m "approximates" f under condition $C \wedge C'$, then $\mathsf{map}(f_m)$ "approximates" $\mathsf{map}(f)$ under condition C, hence it suffices to take $m_{\mathsf{map}(f)} \stackrel{\text{def}}{=} m_f$.

For the case $\mathsf{if}\ f_1\ \mathsf{then}\ f_2\ \mathsf{else}\ f_3$, we start by applying induction hypothesis to $C \Rightarrow (f_1(A) \Downarrow)$. If its complexity is exponential, we are done. Else, it will evaluate to an abstract expression of type $\mathbb{B}$, which, for all practical purposes, is equivalent to some abstract expression of the form $\mathsf{if}\ C'\ \mathsf{then}\ \mathsf{true}\ \mathsf{else}\ \mathsf{false}$. Now we apply induction hypothesis to compute $C \wedge C' \Rightarrow (f_2(A) \Downarrow)$ and $C \wedge \mathsf{not}(C') \Rightarrow (f_3(A) \Downarrow)$. If any of them has exponential complexity, then so has the whole if-expression, and we are done. So assume that they evaluate to A' and A'' respectively. Then $C \Rightarrow ((\mathsf{if}\ f_1\ \mathsf{then}\ f_2\ \mathsf{else}\ f_3)(A) \Downarrow \mathsf{if}\ C\ \mathsf{then}\ A'\ \mathsf{else}\ A'')$.

Note how the condition C is enriched during the induction, both in the $\mathsf{map}(f)$ case and in the if-case.

We devote the rest of the proof to powerset case. We only look at the case when the abstract expression is of the form $\{A(\vec{x}, \vec{y}) \mid \vec{x} = 0, \mathsf{n}; C'(\vec{x}, \vec{y})\}$ (the other two cases, $A \cup A'$ or the conditional, are reduced to this one). We handle $\mathsf{powerset}(\{A(\vec{x}, \vec{y}) \mid \vec{x} = 0, \mathsf{n}; C'(\vec{x}, \vec{y})\})$ by proving that one of the following cases must occur:

1. There is some number m, independent of n (dependent only on C and A), such for any $n \geq 0$ and for any $\rho \in \mathsf{Env}^{[n]}$ for which $[\![C(\vec{y})]\!]^{[n]}(\rho) = \mathsf{T}$, the set $[\![\{A(\vec{x}, \vec{y}) \mid \vec{x} = 0, \mathsf{n}; C'(\vec{x}, \vec{y})\}]\!]^{[n]}(\rho)$ has at most m elements. More, we will show that in this case we can actually find abstract expressions $A_1(\vec{y}), \ldots, A_m(\vec{y})$ naming these at most m elements. In this case $\mathsf{powerset}(\{A(\vec{x}, \vec{y}) \mid \vec{x} = 0, \mathsf{n}; C'(\vec{x}, \vec{y})\}) \Downarrow A'$, were A' is an abstract expression enumerating all 2^m subsets of $\{A_1(\vec{y}), \ldots, A_m(\vec{y})\}$. Obviously, in this case f is equivalent to the m^{th} approximation of powerset, i.e. $\mathsf{powerset}_m(\{A(\vec{x}, \vec{y}) \mid \vec{x} = 0, \mathsf{n}; C'(\vec{x}, \vec{y})\}) \Downarrow A'$. Then item 1 of Theorem 7.17 holds.
2. For every n, there is some environment $\rho \in \mathsf{Env}^{[n]}$ such that $[\![C(\vec{y})]\!]^{[n]}(\rho) = \mathsf{T}$, such that the set $[\![\{A(\vec{x}, \vec{y}) \mid \vec{x} = 0, \mathsf{n}; C'(\vec{x}, \vec{y})\}]\!]^{[n]}(\rho)$ contains at least $\Omega(n)$ distinct elements. Then item 2 of Theorem 7.17 holds.

Suppose the first condition does not hold, i.e. for any $m \geq 0$, we can find $n \geq 0$ and $\rho \in \mathsf{Env}^{[n]}$, such that $[\![C]\!]^{[n]}(\rho) = \mathsf{T}$, and $\mathsf{card}([\![\{A(\vec{x}, \vec{y}) \mid \vec{x} = 0, \mathsf{n}; C'(\vec{x}, \vec{y})\}]\!]^{[n]}(\rho)) \geq m$. Let $\vec{x}'$ be fresh variables. Recall that equality is definable at all types in $\mathcal{NRA}$ (Example 2.5). Then we "evaluate" $= (\langle A(\vec{x}, \vec{y}), A(\vec{x}', \vec{y})\rangle) \Downarrow$, using Theorem 5.2, and get an abstract expression $A'(\vec{x}, \vec{x}', \vec{y})$ of type $\mathbb{B}$ such that $= (\langle A(\vec{x}, \vec{y}), A(\vec{x}', \vec{y})\rangle) \Downarrow A'(\vec{x}, \vec{x}', \vec{y})$. Since $A'(\vec{x}, \vec{x}', \vec{y})$ is of type $\mathbb{B}$, it has to be equivalent to an expression of the form $\mathsf{if}\ E(\vec{x}, \vec{x}', \vec{y})\ \mathsf{then}\ \mathsf{true}\ \mathsf{else}\ \mathsf{false}$. Intuitively, $E(\vec{x}, \vec{x}', \vec{y})$ is true iff $\vec{A}(\vec{x}, \vec{y})$ and $\vec{A}(\vec{x}', \vec{y})$ are equal. Consider now the condition $D(\vec{x}, \vec{x}', \vec{y}) \stackrel{\text{def}}{=} C(\vec{y}) \wedge C'(\vec{x}, \vec{y}) \wedge C'(\vec{x}', \vec{y}) \wedge \mathsf{not}\ E(\vec{x}, \vec{x}', \vec{y})$: for any $m > 0$, there is some $\vec{y}$ and a clique of length m for D at $\vec{y}$. For D we apply the Clique Property (Proposition 7.16), and get an affine a.e. U, such that for any ρ, if $[\![\vec{y}]\!]^{[n]}(\rho) \in U$, then we have (1) $[\![C]\!]^{[n]}(\rho) = \mathsf{T}$, and (2) $[\![\{A(\vec{x}, \vec{y}) \mid \vec{x} = 0, \mathsf{n}; C'(\vec{x}, \vec{y})\}]\!]^{[n]}(\rho)$ has $\Omega(n)$ elements. Therefore $\mathsf{powerset}([\![\{A(\vec{x}, \vec{y}) \mid \vec{x} = 0, \mathsf{n}; C'(\vec{x}, \vec{y})\}]\!]^{[n]})$ has $\Omega(2^{cn})$ elements. Hence, the complexity of $C \Rightarrow (f(\{A(\vec{x}, \vec{y}) \mid \vec{x} = 0, \mathsf{n}; C'(\vec{x}, \vec{y})\})) \Downarrow$ is $\Omega(2^{cn})$.

It remains to prove that, in Case 1, we can actually find m abstract expressions denoting the at most m elements of the set $\{A(\vec{x}, \vec{y}) \mid \vec{x} = 0, \mathsf{n}; C'(\vec{x}, \vec{y})\}$. Consider the condition $E(\vec{x}, \vec{x}', \vec{y})$ as above, stating that $A(\vec{x}, \vec{y})$ and $A(\vec{x}', \vec{y})$ are equal. Consider now $m+1$ fresh variable tuples, $\vec{x}^1, \ldots, \vec{x}^m, \vec{x}$. By the quantifier elimination Proposition 5.1, we find a condition $D(\vec{x}^1, \ldots, \vec{x}^m, \vec{y})$ equivalent to:

$$\forall \vec{x}.(C(\vec{x}, \vec{y}) \rightarrow E(\vec{x}^1, \vec{x}, \vec{y}) \vee \ldots \vee E(\vec{x}^m, \vec{x}, \vec{y}))$$

Intuitively, D states that $A(\vec{x}^1, \vec{y}), \ldots, A(\vec{x}^m, \vec{y})$ cover all elements of the set. Next we need the following Lemma, stating that Skolem functions are expressible as abstract expressions:

Lemma 7.18 (Skolemization Lemma) *Let $C(x,\vec{y})$ be some condition, $\vec{y} = \langle y_1, \ldots, y_q\rangle$. Then there exists some abstract expression $S(\vec{y})$ such that $\forall n \geq 0, \forall \vec{y} \in [n]^q$ if $\exists x.C(x,\vec{y})$ is true, then $C(S(\vec{y}),\vec{y})$ is true. More precisely: $\forall n \geq 0$, $\forall \rho \in \mathsf{Env}^{[n]}$, if there exists some $c \in [n]$ such that $[\![C(x,\vec{y})]\!]^{[n]}(\rho[x = c]) = \mathsf{T}$, then $[\![C(x,\vec{y})]\!]^{[n]}(\rho[x = [\![S]\!]^{[n]}(\vec{y})(\rho)]) = \mathsf{T}$.*

Proof. (Sketch) Let C be in disjunctive normal form, $C = C_1 \vee \ldots \vee C_m$. Suppose we find Skolem functions $S_1, \ldots, S_m$ for the conjunctive conditions $C_1, \ldots, C_m$; then the Skolem function for C is:

$$\begin{array}{rcl} S(\vec{y}) & \stackrel{\text{def}}{=} & \mathsf{if}\ \exists x.C_1(x,\vec{y})\ \mathsf{then}\ S_1(\vec{y}) \\ & & \mathsf{else\ if}\ \exists x.C_2(x,\vec{y})\ \mathsf{then}\ S_2(\vec{y}) \\ & & \mathsf{else}\ \ldots \\ & & \mathsf{else\ if}\ \exists x.C_m(x,\vec{y})\ \mathsf{then}\ S_m(\vec{y}) \\ & & \mathsf{else}\ 0 \end{array}$$

This is not quite an abstract expression, but can be translated into one, by quantifier elimination, Proposition 5.1. So without loss of generality we may assume that C is a conjunction of simple conditions. In that case, if $C(x,\vec{y})$ has at least one positive condition on x, say $x = y_i + c$, then we pick $S(\vec{y}) \stackrel{\text{def}}{=} y_i + c$. Else, let $C'(x,\vec{y}) \stackrel{\text{def}}{=} (x \neq A_1 \wedge \ldots x \neq A_p)$ be all simple conditions of C mentioning x. We build S as follows:

$$\begin{array}{rcl} S(\vec{y}) & \stackrel{\text{def}}{=} & \mathsf{if}\ C'(0,\vec{y})\ \mathsf{then}\ 0 \\ & & \mathsf{else\ if}\ (\mathsf{n} \geq 1) \wedge C'(1,\vec{y})\ \mathsf{then}\ 1 \\ & & \mathsf{else\ if}\ (\mathsf{n} \geq 2) \wedge C'(2,\vec{y})\ \mathsf{then}\ 2 \\ & & \mathsf{else}\ \ldots \\ & & \mathsf{else\ if}\ (\mathsf{n} \geq p) \wedge C'(p,\vec{y})\ \mathsf{then}\ p \\ & & \mathsf{else}\ 0 \end{array}$$

□

So now we conclude the proof of Theorem 7.17, by applying repeatedly the Skolemization Lemma to the condition:

$$\exists \vec{x}^1.\exists \vec{x}^2.\ldots.\exists \vec{x}^m.(C(\vec{y}) \Rightarrow D(\vec{x}^1, \ldots, \vec{x}^m, \vec{y}))$$

which is true $\forall n \geq 0, \forall \rho \in \mathsf{Env}^{[n]}$. We get the m Skolem functions $\vec{S}_1(\vec{y}), \ldots, \vec{S}_m(\vec{y})$ s.t. $\forall n \geq 0$, $\forall \rho \in \mathsf{Env}^{[n]}$, if $[\![C]\!]^{[n]}(\rho) = \mathsf{T}$, then, by denoting $\vec{c}^1 \stackrel{\text{def}}{=} [\![S_1]\!]^{[n]}(\rho), \ldots, \vec{c}^m \stackrel{\text{def}}{=} [\![S_m]\!]^{[n]}(\rho)$, we have

$$[\![D]\!]^{[n]}(\rho[\vec{x}^1 = \vec{c}^1, \ldots, \vec{x}^m = \vec{c}^m]) = \mathsf{T}$$

We conclude that $A_1(\vec{y}) \stackrel{\text{def}}{=} A(\vec{S}_1(\vec{y}),\vec{y}), \ldots, A_m(\vec{y}) \stackrel{\text{def}}{=} A(\vec{S}_m(\vec{y}),\vec{y})$ are abstract expressions denoting *all* elements of the set (maybe with duplicates).

□

Now we are ready to prove our main results, Theorem 3.1 and Proposition 3.2, which we restate here and prove together:

Theorem 3.1 *For any function $f \in \mathcal{NRA}(\mathsf{powerset})$ of type $f : \{\mathbb{Z} \times \mathbb{Z}\} \to \{\mathbb{Z} \times \mathbb{Z}\}$ such that $f(r_n) \Downarrow q_n$ for every $n \geq 0$, the complexity of $f(r_n) \Downarrow$ is $\Omega(2^{cn})$, for some $c > 0$.*

Proposition 3.2 *For any type t and any function $f : \{\mathbb{Z} \times \mathbb{Z}\} \to t$ in $\mathcal{NRA}(\mathsf{powerset})$, either there exists in $\mathcal{NRA}$ some approximation f_m of f such that $f_m(r_n) = f(r_n)$, $\forall n \geq 0$, or the complexity of $f(r_n) \Downarrow$ is $\Omega(2^{cn})$, for some $c > 0$.*

Proof. Let $f \in \mathcal{NRA}(\mathsf{powerset})$, $f : \{\mathbb{Z} \times \mathbb{Z}\} \to t$ be such that the complexity of $f(r_n) \Downarrow$ is not $\Omega(2^{cn})$. Taking $C = \mathsf{true}$ in Theorem 7.17, only case 1 can hold, hence there is some approximation f_m of f performing the same computation on r_n (which proves Proposition 3.2), and there is some abstract expression denoting its result; by Proposition 6.12, its result cannot be $\mathsf{tc}(r_n)$, which proves Theorem 3.1. □

Proposition 3.3 *Any expression in $\mathcal{NRA}(\mathsf{powerset})$ for computing* parity *will have exponential complexity.*

Proof. Let $f \in \mathcal{NRA}(\mathsf{powerset})$, of type $f : \{\mathbb{Z}\} \to \mathbb{B}$ computing parity without exponential complexity and consider the abstract expression $A = \{x \mid x = 0, \mathsf{n}; \}$. Then, by Theorem 7.17 there exists an abstract expression A' s.t. $f(A) \Downarrow A'$. Since A' is closed and of type $\mathbb{B}$, it has to be essentially equivalent to an expression of the form if C then true else false. But condition C is closed, so it is either equivalent to true for n large enough, or equivalent to false for n large enough. In neither case can f denote parity. □

8 Conclusions and Further Research

In this paper we proved that any expression in $\mathcal{NRA}(\mathsf{powerset})$ needs exponential space to compute transitive closure, under the straightforward, naive evaluation strategy. We did this by proving that exponential space is needed to compute the transitive closure of a very special kind of relation, namely a chain of length n. In particular this implies the stronger result that *deterministic* transitive closure (i.e. transitive closure of a graph whose nodes have outdegree ≤ 1, see [Imm87]) is not efficiently expressible in $\mathcal{NRA}(\mathsf{powerset})$.

We conjecture that our result generalizes in that any query expressible in an efficient way in $\mathcal{NRA}(\mathsf{powerset})$ is already expressible in $\mathcal{NRA}$. However, our techniques do not apply directly to the more general result, as the Example given in Section 3 suggests.

Our results strongly depend on (1) the complexity measure $\mathsf{size}(C)$ on objects (see Subsection 2.1), and (2) the evaluation strategy $f(C) \Downarrow C'$ (see Figure 3). Artificial evaluation strategies may be conceived for which transitive closure can be expressed in polynomial time in $\mathcal{NRA}(\mathsf{powerset})$. E.g. consider the "artificial" evaluation strategy $f(C) \Downarrow_a C'$ defined by: "if f is the expression denoting transitive closure of Example 2.10, then evaluate $\mathsf{tc}(C)$ using a polynomial time algorithm; else evaluate $f(C) \Downarrow C'$ using the definition of Figure 3". Abiteboul and Hillebrand [AH95] define a more natural evaluation strategy which computes queries with set height ≤ 2 in a pipeline fashion, and computes other queries using $\Downarrow$. Then the transitive-closure expression of Example 2.10 will be computed in polynomial space, under this strategy (but not in polynomial time).

Finally, we make some observation related to the complexity of the class of queries which are expressible in $\mathcal{NRA}(\mathsf{powerset})$ with a polynomial complexity. Recall that the class AC^0 is the class of functions $f : \{0,1\}^* \to \{0,1\}^*$ computable by a "uniform" family of circuits made of NOT gates and unbounded fan-in AND and OR gates, having polynomial size and constant depth (see [BIS90]). The class TC^0 is defined similarly, but by allowing an additional type of gates, the *threshold gates*; a threshold gate with number k will output an 1 iff at least k of its inputs are 1.

It is known that $\mathcal{NRA} \subseteq AC^0$ [SBT94]. Consider now the *PTIME*-fragment of $\mathcal{NRA}(\mathsf{powerset})$; more formally:

Definition 8.1 *Let*

$$\mathcal{NRA}(\mathsf{powerset})\mid_{PTIME} \stackrel{\text{def}}{=} \begin{array}{l}\{f \mid f \in \mathcal{NRA}(\mathsf{powerset}),\\ \exists P,\ P \textit{ is a polynomial},\\ \forall C.\mathsf{complex}(f, C) \leq P(\mathsf{size}(C))\}\end{array}$$

Then we can prove:

Proposition 8.2 $\mathcal{NRA}(\mathsf{powerset})\mid_{PTIME} \subseteq TC^0$.

Since transitive closure is complete for *NLOGSPACE* [Imm87] it follows that $TC^0 \neq$ *NLOGSPACE* implies a weaker form of our result, namely that transitive closure is not in $\mathcal{NRA}(\mathsf{powerset})_{PTIME}$. However, TC^0 and *NLOGSPACE* have not been separated yet.

Acknowledgments

We would like to thank Val Tannen, Peter Buneman, and Jan Van den Bussche for their suggestions and support, Bart Kuijpers for carefully reading and commenting on the paper, and the reviewers for their useful comments.

References

[AB88] Serge Abiteboul and Catriel Beeri. On the power of languages for the manipulation of complex objects. In *Proceedings of International Workshop on Theory and Applications of Nested Relations and Complex Objects*, Darmstadt, 1988. Also available as INRIA Technical Report 846.

[AH95] S. Abiteboul and G. Hillebrand. Space usage in functional query languagess. In G. Gottlob and M. Vardi, editors, *Database Theory — ICDT'95*, pages 439–454, Prague, Czech Republic, January 1995. Springer Verlag.

[AU79] Alfred V. Aho and Jeffrey D. Ullman. Universality of data retrieval languages. In *Proceedings 6th Symposium on Principles of Programming Languages, Texas, January 1979*, pages 110–120, 1979.

[AV91] Serge Abiteboul and Victor Vianu. Generic computation and its complexity. In *Proceedings of 23rd ACM Symposium on the Theory of Computing*, 1991.

[BBW92] Val Breazu-Tannen, Peter Buneman, and Limsoon Wong. Naturally embedded query languages. In J. Biskup and R. Hull, editors, *LNCS 646: Proceedings of 4th International Conference on Database Theory, Berlin, Germany, October, 1992*, pages 140–154. Springer-Verlag, October 1992. Available as UPenn Technical Report MS-CIS-92-47.

[BIS90] David Mix Barrington, Neil Immerman, and Howard Straubing. On uniformity within NC^1. *Journal of Computer and System Sciences*, 41:274–306, 1990.

[BM75] Donald W. Barnes and John M. Mack. *An Algebraic Introduction to Mathematical Logic.* Springer Verlag, 1975.

[Bol79] Bela Bollobas. *Graph Theory: An Introductory Course.* Springer-Verlag, 1979.

[CH80] Ashok Chandra and David Harel. Computable queries for relational databases. *Journal of Computer and System Sciences*, 21(2):156–178, 1980.

[dB94] Jan Van den Bussche. Personal communication, 1994.

[HS91] Richard Hull and Jianwen Su. On the expressive power of database queries with intermediate types. *Journal of Computer and System Sciences*, 43:219–267, 1991.

[Imm87] Neil Immerman. Languages that capture complexity classes. *SIAM Journal of Computing*, 16:760–778, 1987.

[Kah87] Gilles Kahn. Natural semantics. In *Proceedings of Symposium on Theoretical Aspects of Computer Science*, pages 22–39. Springer-Verlag, 1987.

[KV93] Gabriel M. Kuper and Moshe Y. Vardi. On the complexity of queries in the logical data model. *Theoretical Computer Science*, 116(1):33–58, August 1993.

[PG88] Jan Paredaens and Dirk Van Gucht. Possibilities and limitations of using flat operators in nested algebra expressions. In *Proceedings of 7th ACM Symposium on Principles of Database Systems,Austin, Texas*, pages 29–38, 1988.

[PG92] Jan Paredaens and Dirk Van Gucht. Converting nested relational algebra expressions into flat algebra expressions. *ACM Transaction on Database Systems*, 17(1):65–93, March 1992.

[SBT94] Dan Suciu and Val Breazu-Tannen. A query language for NC. In *Proceedings of 13th ACM Symposium on Principles of Database Systems*, pages 167–178, Minneapolis, Minnesota, May 1994. See also UPenn Technical Report MS-CIS-94-05.

[SS86] H.-J. Schek and M. H. Scholl. The relational model with relation-valued attributes. *Information Systems*, 11(2):137–147, 1986.

[TF86] S. J. Thomas and P. C. Fischer. Nested relational structures. In P. C. Kanellakis and F. P. Preparata, editors, *Advances in Computing Research: The Theory of Databases*, pages 269–307, London, England, 1986. JAI Press.

[Won92] Limsoon Wong. A conservative property of a nested relational query language. Technical Report MS-CIS-92-59/L&C 48, University of Pennsylvania, Philadelphia, Pennsylvania 19104, July 1992.

[Won93] Limsoon Wong. Normal forms and conservative properties for query languages over collection types. In *Proceedings of 12th ACM Symposium on Principles of Database Systems*, pages 26–36, Washington, D. C., May 1993. See also UPenn Technical Report MS-CIS-92-59.