Abstractions. A fruitful idea in the analysis of complex systems is that of abstractions: Instead of working with the original model, one considers a model that has a simpler structure but preserves pertinent properties.

A prominent example of such an abstraction is the Parikh image, which is available whenever the semantics of a model is given as a formal language. If $L$ is a language over an alphabet $X = \{x_1, \ldots, x_n\}$, then its Parikh image $\Psi(L)$ consists of all vectors $(a_1, \ldots, a_n) \in \mathbb{N}^n$ such that there is a $w \in L$ in which $x_i$ occurs $a_i$ times for $i \in \{1, \ldots, n\}$. The Parikh image is a useful abstraction, provided that for the languages $L$ at hand, $\Psi(L)$ can be described using simpler means than $L$ itself. This means, most applications of Parikh images are confined to situations where $\Psi(L)$ is semilinear (or, equivalently, definable in Presburger arithmetic). For example, this is the case for context-free languages \cite{10}, languages accepted by blind (or reversal-bounded) counter automata \cite{7}, and combinations thereof \cite{1,4,6}.

Of course, there are important types of languages that do not guarantee semilinearity of their Parikh image. For example, Parikh images of languages accepted by higher-order pushdown automata are not semilinear in general: It is easy to construct a second-order pushdown automaton for the language $\{a^{2^n} \mid n \geq 0\}$.

Downward closures. However, there is an abstraction of languages that guarantees a simple description for every language whatsoever and still reflects important properties of the abstracted language—the downward closure. For words $u, v \in X^*$, we write $u \preceq v$ if $u = u_1 \cdots u_n$ and $v = v_0u_1v_1 \cdots u_nv_n$ for some $u_1, \ldots, u_n, v_0, \ldots, v_n \in X^*$. Then, the downward closure of $L$ is defined as

$$L_\downarrow = \{ u \in X^* \mid \exists v \in L : u \preceq v \}.$$  

In other words, the downward closure consists of the set of all (not necessarily contiguous) subwords of members of $L$. It has the remarkable property that it is regular for every set of words $L \subseteq X^*$, which is due to the fact that the subword ordering $\preceq$ is a well-quasi-ordering \cite{5}.

Moreover, downward closures preserve useful information about the abstracted language. Suppose $L$ describes the behavior of a system that is observed through a lossy channel, meaning that on the way to the observer, arbitrary actions can get lost. Then, $L_\downarrow$ is the set of words received by the observer \cite{3}. Hence, given the downward closure as a finite automaton, we can decide whether two systems are equivalent under such observations, and even whether one system includes the behavior of another.

However, while we know that for every $L$, there exists a finite automaton for $L_\downarrow$, it is not clear in general how to compute them. In fact, there are examples of languages classes for which downward closures are known not to be computable \cite{2,9}. For an overview of the language classes for which computability is known, see \cite{11}.
Recently, a new approach to the computation of downward closures has been proposed. For a wide range of language classes, the approach reduces the computation to the \textit{simultaneous unboundedness problem (SUP)} \cite{Zetzsche15}. The latter asks, given a language \( L \subseteq a_1^* \cdots a_n^* \), whether for every \( k \in \mathbb{N} \), there are \( x_1, \ldots, x_n \geq k \) with \( a_1^{x_1} \cdots a_n^{x_n} \in L \). In fact, this result can be used to compute downward closures for indexed languages. Since these coincide with those languages accepted by second-order pushdown automata \cite{Maslov76}, we have the following.

**Theorem 1.** Downward closures are computable for second-order pushdown automata.

In this talk, I will sketch the method for computing downward closures of indexed languages.

\textbf{References}

\begin{enumerate}
\end{enumerate}

\textit{Technische Universität Kaiserslautern, Fachbereich Informatik, Concurrency Theory Group}

\textit{E-mail address: zetzsche@cs.uni-kl.de}