FSFMA 2013

16th July 2013
Singapore

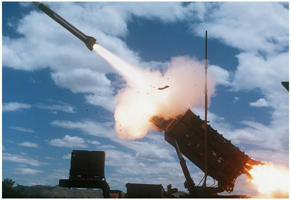
Dynamic Clock Elimination in Parametric Timed Automata

Étienne André

Laboratoire d'Informatique de Paris Nord
Université Paris 13, Sorbonne Paris Cité

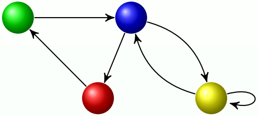
Context: Verifying Complex Timed Systems (1/2)

- Need for early bug detection
  - Bugs discovered when final testing: expensive
  - Need for thorough modeling and verification



Context: Verifying Complex Timed Systems (2/2)

- Use formal methods



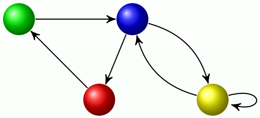
A finite model of the system

$AG\neg$●

A formula to be satisfied

Context: Verifying Complex Timed Systems (2/2)

- Use formal methods



$\overset{?}{\models}$

$AG\neg$●

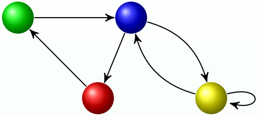
A finite model of the system

A formula to be satisfied

- Question: does the model of the system satisfy the formula?

Context: Verifying Complex Timed Systems (2/2)

- Use formal methods



$\overset{?}{\models}$

$AG\neg$●

A finite model of the system

A formula to be satisfied

- Question: does the model of the system satisfy the formula?

Yes

No

Counterexample

Context: Parameter Synthesis

- Timed systems are characterized by a set of timing constants
  - "The packet transmission lasts for $50\,ms$"
  - "The sensor reads the value every $10\,s$"
  - etc.
- Verification for one set of constants does not guarantee the correctness for other values
- Challenges
  - Numerous verifications: is the system correct for any value within $[40; 60]$?
  - Optimization: until what value can we increase 10?
  - Robustness: What happens if 50 is implemented with 49.99?

Context: Parameter Synthesis

- Timed systems are characterized by a set of timing constants
    - "The packet transmission lasts for $50\,ms$"
    - "The sensor reads the value every $10\,s$"
    - etc.
- Verification for one set of constants does not guarantee the correctness for other values
- Challenges
    - Numerous verifications: is the system correct for any value within $[40;60]$?
    - Optimization: until what value can we increase 10?
    - Robustness: What happens if 50 is implemented with 49.99?
- Parameter synthesis
    - Consider that timing constants are unknown constants (parameters)
    - Find good values for the parameters

Outline

1. Parametric Timed Automata
2. Motivation: Clock Reduction
3. Dynamic Elimination
4. Experimental Validation
5. Conclusion

Outline

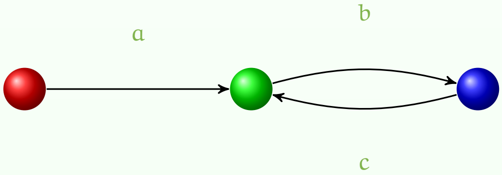
1. Parametric Timed Automata
2. Motivation: Clock Reduction
3. Dynamic Elimination
4. Experimental Validation
5. Conclusion

Timed Automaton

- Finite state automaton (sets of locations)

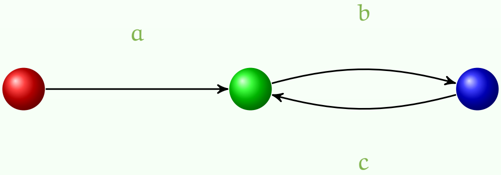
Timed Automaton

- Finite state automaton (sets of locations and actions)



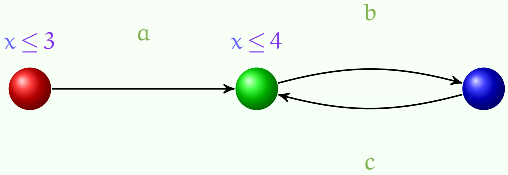
Timed Automaton

- Finite state automaton (sets of locations and actions) augmented with
  - A set X of clocks (i.e., real-valued variables evolving linearly at the same rate [Alur and Dill, 1994])



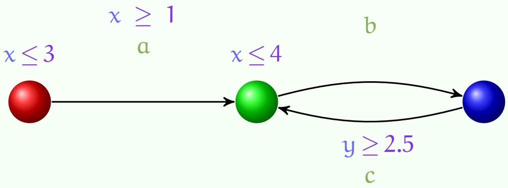
Timed Automaton

- Finite state automaton (sets of locations and actions) augmented with
  - A set X of clocks (i.e., real-valued variables evolving linearly at the same rate [Alur and Dill, 1994])
- Features
  - Location invariant: property to be verified to stay at a location



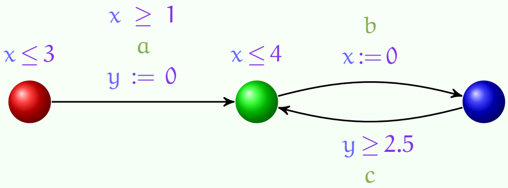
Timed Automaton

- Finite state automaton (sets of locations and actions) augmented with
  - A set X of clocks (i.e., real-valued variables evolving linearly at the same rate [Alur and Dill, 1994])
- Features
  - Location invariant: property to be verified to stay at a location
  - Transition guard: property to be verified to enable a transition



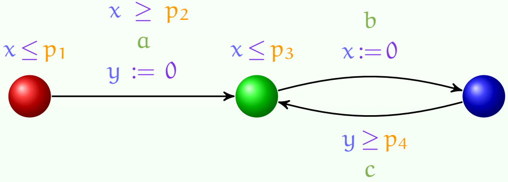
Timed Automaton

- Finite state automaton (sets of locations and actions) augmented with
  - A set X of clocks (i.e., real-valued variables evolving linearly at the same rate [Alur and Dill, 1994])
- Features
  - Location invariant: property to be verified to stay at a location
  - Transition guard: property to be verified to enable a transition
  - Clock reset: some of the clocks can be set to 0 at each transition



Parametric Timed Automaton (PTA)

- Finite state automaton (sets of locations and actions) augmented with
  - A set X of clocks (i.e., real-valued variables evolving linearly at the same rate [Alur and Dill, 1994])
  - A set P of parameters (i.e., unknown constants), used in guards and invariants [Alur et al., 1993]
- Features
  - Location invariant: property to be verified to stay at a location
  - Transition guard: property to be verified to enable a transition
  - Clock reset: some of the clocks can be set to 0 at each transition



Semantics of Parametric Timed Automata

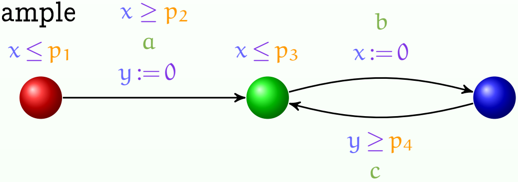
- State of a PTA: couple (q, C), where
  - q is a location,
  - C is a constraint (conjunction of inequalities) over X and P

Semantics of Parametric Timed Automata

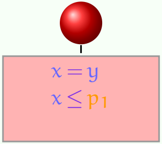
- State of a PTA: couple (q, C), where
    - q is a location,
    - C is a constraint (conjunction of inequalities) over X and P
- Path: alternating sequence of states and actions

Semantics of Parametric Timed Automata

- State of a PTA: couple (q, C), where
  - q is a location,
  - C is a constraint (conjunction of inequalities) over X and P
- Path: alternating sequence of states and actions
- Example

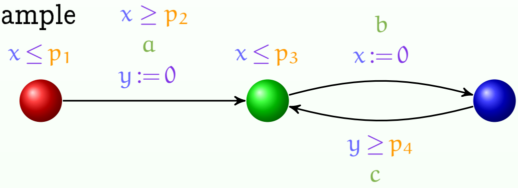


  - Possible path for this PTA

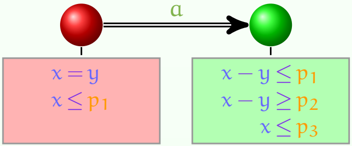


Semantics of Parametric Timed Automata

- State of a PTA: couple (q, C), where
  - q is a location,
  - C is a constraint (conjunction of inequalities) over X and P
- Path: alternating sequence of states and actions
- Example

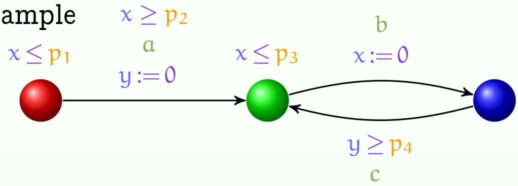


  - Possible path for this PTA

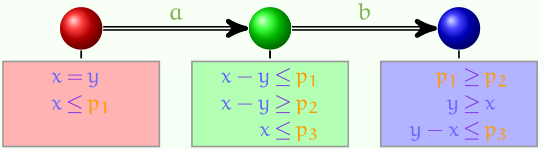


Semantics of Parametric Timed Automata

- State of a PTA: couple (q, C), where
  - q is a location,
  - C is a constraint (conjunction of inequalities) over X and P
- Path: alternating sequence of states and actions
- Example

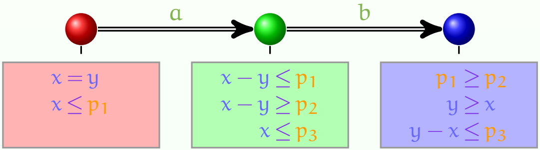


  - Possible path for this PTA



Traces

- Trace over a PTA: time-abstract path
  - Finite alternating sequence of locations and actions



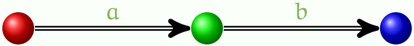
$x = y$
$x \leq p_1$

$x - y \leq p_1$
$x - y \geq p_2$
$x \leq p_3$

$p_1 \geq p_2$
$y \geq x$
$y - x \leq p_3$

Traces

- Trace over a PTA: time-abstract path
    - Finite alternating sequence of locations and actions



Outline

1. Parametric Timed Automata
2. Motivation: Clock Reduction
3. Dynamic Elimination
4. Experimental Validation
5. Conclusion

Reducing the number of clocks

- The fewer clocks, the more efficient model checking is [Bengtsson and Yi, 2003]
- Consequence: State space reduction
  - Smaller constraints (represented as arrays, matrices, etc.)
  - Less states (due to side-effect merging)
- Clock reduction native in some formalisms
  - ☺ Parametric time Petri nets [Traonouez et al., 2009]
  - ☺ Parametric stateful timed CSP [Sun et al., 2013]

Reducing the number of clocks

- The fewer clocks, the more efficient model checking is [Bengtsson and Yi, 2003]
- Consequence: State space reduction
  - Smaller constraints (represented as arrays, matrices, etc.)
  - Less states (due to side-effect merging)
- Clock reduction native in some formalisms
  - ☺ Parametric time Petri nets [Traonouez et al., 2009]
  - ☺ Parametric stateful timed CSP [Sun et al., 2013]
  - ☹ . . . but not in PTA

Reducing the number of clocks: Some approaches

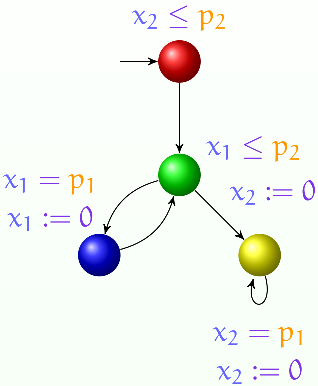
- Clock elimination in non-parametric timed automata [Daws and Yovine, 1996]
    - Detection of (in)active clocks
    - Detection of clocks equal to each other
    - Relatively easy in a non-parametric setting (use of Difference Bound Matrices)
- (Tentative) elimination of the global clocks in a network of timed automata in a distributed setting [Balaguer and Chatain, 2012]
- Native elimination in other formalisms [Traonouez et al., 2009, Sun et al., 2013]
    - Translation from timed automata?

Outline

1. Parametric Timed Automata
2. Motivation: Clock Reduction
3. **Dynamic Elimination**
4. Experimental Validation
5. Conclusion

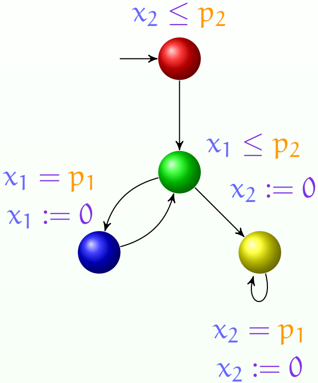
Principle

- Inactive clocks
  - In 🔵, the value of x_2 is useless. It will not be used until its next reset when entering 🟡.



Principle

- Inactive clocks
  - In ●, the value of x_2 is useless. It will not be used until its next reset when entering ●.
- Goal: detect and eliminate inactive clocks
  - ⇒ Smaller memory
  - ⇒ Less states
  - ⇒ Better termination



Assumptions

Remark

Detecting really useless clocks would require us to know the future, hence to perform the analysis… which we want to avoid

Assumptions

Remark

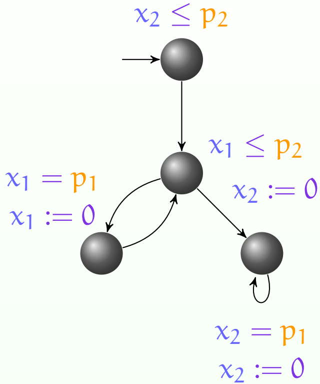
Detecting really useless clocks would require us to know the future, hence to perform the analysis… which we want to avoid

- Assumptions
  - Static a priori detection of the useless clocks
  - Local clocks only
  - Dynamic elimination during the analysis
- Consequence: possible under-approximation of the set of eliminated clocks

Static Detection

- Backward marking algorithm for a clock x
    - Goal: mark all locations where x is useful
    - Start by marking the locations where x is used (invariant or outgoing guard)
    - Iterate in a backward manner until a reset is found
    - Stop when reaching fixpoint

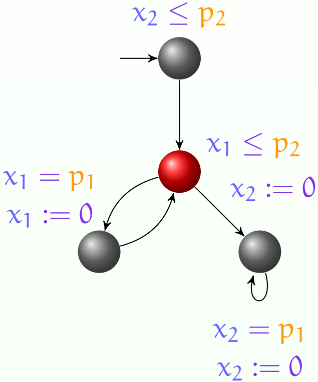
Example for x_1:



Static Detection

- Backward marking algorithm for a clock x
  - Goal: mark all locations where x is useful
  - Start by marking the locations where x is used (invariant or outgoing guard)
  - Iterate in a backward manner until a reset is found
  - Stop when reaching fixpoint

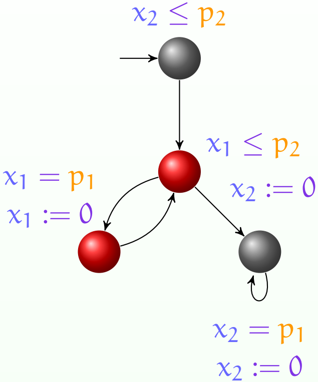
Example for x_1:



Static Detection

- Backward marking algorithm for a clock x
  - Goal: mark all locations where x is useful
  - Start by marking the locations where x is used (invariant or outgoing guard)
  - Iterate in a backward manner until a reset is found
  - Stop when reaching fixpoint

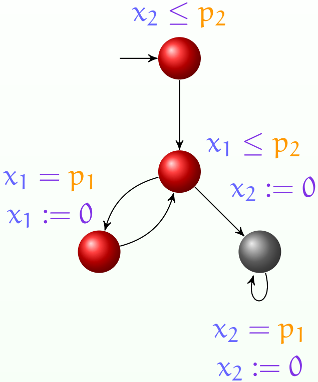
Example for x_1:



Static Detection

- Backward marking algorithm for a clock x
    - Goal: mark all locations where x is useful
    - Start by marking the locations where x is used (invariant or outgoing guard)
    - Iterate in a backward manner until a reset is found
    - Stop when reaching fixpoint

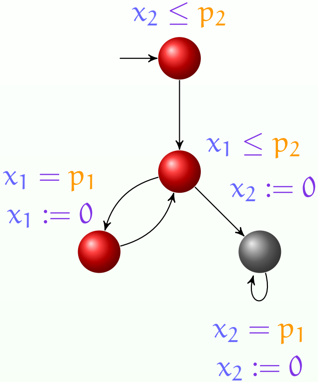
Example for x_1:



Static Detection

- Backward marking algorithm for a clock x
    - Goal: mark all locations where x is useful
    - Start by marking the locations where x is used (invariant or outgoing guard)
    - Iterate in a backward manner until a reset is found
    - Stop when reaching fixpoint

Example for x_1:



Hence, x_1 can be eliminated when in ● .

Dynamic Elimination

- Clocks can be eliminated on-the-fly when computing a new state
  - Refer to the static table of the useless clocks in the current location

- Elimination à la Fourier-Motzkin [Schrijver, 1986]
  - So as not to modify the relationship between other clocks and parameters
    Costly operation

    Example: $x_1 \leq x_2 \leq p_2$ becomes $x_1 \leq p_2$ after elimination of x_2

Characterization

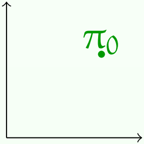
- Bijection between the sets of traces without and with elimination of the clocks
    - All linear-time properties (LTL) can be checked using this optimization
    - The inverse method can be applied [André and Soulat, 2013]
- Bijection between the sets of parametric paths without and with elimination of the clocks
    - Optimization suitable to perform parametric model checking

Outline

IMITATOR

- IMITATOR 2.6 [André et al., 2012]
  - "Inverse Method for Inferring Time AbstracT BehaviOR"
  - 10 000 lines of OCaml code
  - Makes use of the PPL library [Bagnara et al., 2008]
  - Available under the GNU-GPL license
  - Now integrated in the CosyVerif platform [André et al., 2013]

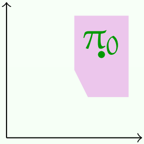
- Experimental validation using the inverse method [André and Soulat, 2013]
  - Generalizes a reference parameter valuation by synthesizing a constraint



IMITATOR

- IMITATOR 2.6 [André et al., 2012]
  - "Inverse Method for Inferring Time AbstracT BehaviOR"
  - 10 000 lines of OCaml code
  - Makes use of the PPL library [Bagnara et al., 2008]
  - Available under the GNU-GPL license
  - Now integrated in the CosyVerif platform [André et al., 2013]

- Experimental validation using the inverse method [André and Soulat, 2013]
  - Generalizes a reference parameter valuation by synthesizing a constraint



Experiments

			IM			IM_{dyn}			Comparison	
Example	$\lvert X\rvert$	$\lvert P\rvert$	$\lvert S\rvert$	$\lvert T\rvert$	t	$\lvert S\rvert$	$\lvert T\rvert$	t	$\lvert S\rvert$	t
Figure 2	2	2	-	-	loop	2	2	0.007	0	0
Figure 3	2	2	-	-	loop	6	8	0.006	0	0
AndOr	4	12	11	11	0.047	11	11	0.050	100	106
SPSMALL	10	26	31	30	0.580	31	30	0.584	100	101
Train	3	6	78	94	0.100	61	76	0.072	78	72
BRP	7	6	429	474	3.50	429	474	3.21	100	92
CSMA/CD$_6$	3	3	13,365	14,271	19.6	13,365	14,271	19.5	100	99
RCP	5	6	327	518	0.68	181	282	0.41	55	60
AAM06	3	8	1,497	1,844	8.28	768	997	2.92	51	35
AM02	3	4	182	215	0.392	182	215	0.386	100	98
BB04	6	7	806	827	25.4	806	827	27.2	100	107
CTC	15	21	1,364	1,363	83.4	201	291	2.52	15	3.0
LA02	3	5	6,290	8,023	710	4,932	7,154	473	78	67
LPPRC10	4	7	78	102	0.375	78	102	0.395	100	105

Sources: `http://www.lsv.ens-cachan.fr/Software/imitator/dynamic/`

Outline

1. Parametric Timed Automata
2. Motivation: Clock Reduction
3. Dynamic Elimination
4. Experimental Validation
5. Conclusion

Conclusion

- Extension of dynamic clock elimination to parametric automata
- Preserves linear-time parametric model checking
- Often leads to state space reduction and memory reduction
- Surprisingly little noticeable overhead, even when the number of clocks remains constant
  - ⇒ Optimization could be added as default in IMITATOR

Perspectives

- Integration of further state space reduction techniques [André et al., 2013]
- Improvement of the internal representation of constraints
  - Relying on the Parma Polyhedra Library [Bagnara et al., 2008]
  - Future work: remove dimensions when eliminating clocks
- Extension to the multi-core setting [Laarman et al., 2013]

Bibliography

References I

Alur, R. and Dill, D. L. (1994).
A theory of timed automata.
Theoretical Computer Science, 126(2):183–235.

Alur, R., Henzinger, T. A., and Vardi, M. Y. (1993).
Parametric real-time reasoning.
In *STOC'93*, pages 592–601. ACM.

André, É., Fribourg, L., Kühne, U., and Soulat, R. (2012).
IMITATOR 2.5: A tool for analyzing robustness in scheduling problems.
In *FM'12*, volume 7436 of *Lecture Notes in Computer Science*, pages 33–36. Springer.

André, É., Fribourg, L., and Soulat, R. (2013).
Merge and conquer: State merging in parametric timed automata.
In *ATVA'13*, Lecture Notes in Computer Science. Springer.

André, E., Hillah, L.-M., Hulin-Hubard, F., Kordon, F., Lembachar, Y., Linard, A., and Petrucci, L. (2013).
CosyVerif: An open source extensible verification environment.
In *ICECCS'13*. IEEE Computer Society.

References II

André, É. and Soulat, R. (2013).
The Inverse Method.
FOCUS Series in Computer Engineering and Information Technology. ISTE Ltd and John Wiley & Sons Inc.

Bagnara, R., Hill, P. M., and Zaffanella, E. (2008).
The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems.
Science of Computer Programming, 72(1–2):3–21.

Balaguer, S. and Chatain, Th. (2012).
Avoiding shared clocks in networks of timed automata.
In *CONCUR'12*, volume 7454 of *Lecture Notes in Computer Science*, pages 100–114. Springer.

Bengtsson, J. and Yi, W. (2003).
Timed automata: Semantics, algorithms and tools.
In *Lectures on Concurrency and Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, pages 87–124. Springer.

Daws, C. and Yovine, S. (1996).
Reducing the number of clock variables of timed automata.
In *RTSS'96*, pages 73–81. IEEE Computer Society.

References III

Laarman, A., Olesen, M. C., Dalsgaard, A. E., Larsen, K. G., and Van De Pol, J. (2013).
Multi-core emptiness checking of timed buchi automata using inclusion abstraction.
In *CAV'13*, volume 8044 of *Lecture Notes in Computer Science*. Springer.

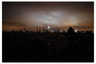
Schrijver, A. (1986).
Theory of linear and integer programming.
John Wiley & Sons, Inc.

Sun, J., Liu, Y., Dong, J. S., Liu, Y., Shi, L., and André, É. (2013).
Modeling and verifying hierarchical real-time systems using Stateful Timed CSP.
ACM Transactions on Software Engineering and Methodology, 22(1):3.1–3.29.

Traonouez, L.-M., Lime, D., and Roux, O. H. (2009).
Parametric model-checking of stopwatch Petri nets.
Journal of Universal Computer Science, 15(17):3273–3304.

Additional explanations

Explanations for the 4 pictures in the beginning



Allusion to the Northeast blackout (USA, 2003)
Computer bug
Consequences: 11 fatalities, huge cost
(Picture actually from the Sandy Hurricane, 2012)

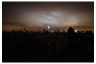
Allusion to any plane crash
(Picture actually from the happy-ending US Airways Flight 1549, 2009)

Allusion to the sinking of the Sleipner A offshore platform (Norway, 1991)
No fatalities
Computer bug: inaccurate finite element analysis modeling
(Picture actually from the Deepwater Horizon Offshore Drilling Platform)

Allusion to the MIM-104 Patriot Missile Failure (Iraq, 1991)
28 fatalities, hundreds of injured
Computer bug: software error (clock drift)
(Picture of an actual MIM-104 Patriot Missile, though not the one of 1991)

Licensing

Source of the pictures used (1/2)



Title: Hurricane Sandy Blackout New York Skyline
Author: David Shankbone
Source: https://commons.wikimedia.org/wiki/File:Hurricane_Sandy_Blackout_New_York_Skyline.JPG
License: CC BY 3.0

Title: Miracle on the Hudson
Author: Janis Krums (cropped by Étienne André)
Source: https://secure.flickr.com/photos/davidwatts1978/3199405401/
License: CC BY 2.0

Title: Deepwater Horizon Offshore Drilling Platform on Fire
Author: ideum
Source: https://secure.flickr.com/photos/ideum/4711481781/
License: CC BY-SA 2.0

Title: DA-SC-88-01663
Author: imcomkorea
Source: https://secure.flickr.com/photos/imcomkorea/3017886760/
License: CC BY-NC-ND 2.0

Source of the pictures used (2/2)

Title: Smiley green alien big eyes (aaah)
Author: LadyofHats
Source: https://commons.wikimedia.org/wiki/File:Smiley_green_alien_big_eyes.svg
License: public domain

Title: Smiley green alien big eyes (cry)
Author: LadyofHats
Source: https://commons.wikimedia.org/wiki/File:Smiley_green_alien_big_eyes.svg
License: public domain

License of this document